UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Reconocimiento de voz basado en características DNN Bottleneck

Máster Universitario en Ingeniería de Telecomunicación

Autor: Irene Martín Calle
Tutor: Alicia Lozano Díez
Ponente: Doroteo Torre Toledano

Junio 2019

Dedicado a quienes confiaron en mí
antes de que yo lo hiciese,
mi familia.

RECONOCIMIENTO DE VOZ BASADO EN CARACTERÍSTICAS DNN BOTTLENECK

AUTOR: Irene Martín Calle
DIRECTOR: Alicia Lozano Díez
PONENTE: Doroteo Torre Toledano

Dpto. de Ingeniería Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio 2019

Resumen

Resumen

En este Trabajo Fin de Master se ha llevado a cabo el desarrollo e implementación de un sistema de reconocimiento automático de voz (Automatic Speech Recognition, ASR) que obtiene de forma automática la transcripción de un segmento de voz determinado. La base de datos empleada para el entrenamiento y la evaluación del sistema es Switchboard-1 Release 2, que pertenece al Consorcio de Datos Lingüísticos (Linguistic Data Consortium, LDC). Por un lado, se ha entrenado un sistema ASR de referencia basado en HMMs (Hidden Markov Models, HMM) utilizando las características acústicas MFCC (Mel Frequency Cepstral Coefficients) con la herramienta Kaldi. Dicha herramienta es una de las más populares en ASR para desarrollar reconocedores de voz y es ampliamente utilizada en el sector industrial y en investigación. Se han entrenado modelos basados en monofonemas ($\Delta + \Delta\Delta$), trifonemas ($\Delta + \Delta\Delta$), LDA + MLLT, SAT, MMI y fMMI además de utilizar un sistema híbrido HMM-DNN, que es una de las técnicas en el estado del arte en reconocimiento del habla. En procesamiento de voz, las Redes Neuronales Profundas (Deep Neural Network, DNN) han mostrado ser capaces de aprender de forma automática una representación de la información contenida en la voz. En este trabajo se representa mediante las características Bottleneck que se utilizan para el entrenamiento del mismo sistema base, extraídas de una DNN implementada en Keras (librería de Python) con la finalidad de reemplazar a las tradicionales características de los enfoques convencionales como los MFCCs. Keras es una de las librerías más poderosas actualmente para evaluar y desarrollar modelos de aprendizaje profundo.

Para evaluar el rendimiento del sistema, se han realizado distintos experimentos cuyos resultados y conclusiones inferidas se reflejan en este trabajo. Se han analizado los resultados obtenidos del sistema basado en características MFCCs y del sistema basado en características Bottleneck. Distintas técnicas como transformaciones del espacio de características o adaptación al locutor han permitido mejorar el rendimiento del sistema inicial, siendo la técnica MMI la que proporciona mejores tasas de error de palabra (WER). Además, el sistema híbrido HMM-DNN, que reemplaza los GMMs en los sistemas tradicionales para el cálculo de los alineamientos, ha permitido mejorar los resultados iniciales aún más. En este escenario experimental, el estudio muestra que las características Bottleneck permiten un rendimiento tan bueno como los MFCCs originales pero no se obtiene una mejora significativa en las tasas de reconocimiento con respecto al sistema basado en MFCCs.

Palabras Clave

Reconocimiento Automático de Voz, Redes Neuronales Profundas, Bottleneck, Modelos Ocultos de Markov, Kaldi, Keras.

Abstract

In this project, the development and implementation of an automatic speech recognition system (ASR), that automatically obtains the transcription of a specific speech segment, has been carried out. The database used for the system training and evaluation is Switchboard-1 Release 2, which belongs to the Linguistic Data Consortium (LDC). On the one hand, a reference ASR system based on HMMs (Hidden Markov Models) has been trained using the acoustic MFCC features (Mel Frequency Cepstral Coefficients) with Kaldi toolkit. This toolkit is one of the most popular in ASR to develop speech recognition systems and is widely used in industrial sector and research. Models based on monophonemes ($\Delta + \Delta\Delta$), triphonemes ($\Delta + \Delta\Delta$), LDA + MLLT, SAT, MMI and fMMI have been trained and also a hybrid system HMM-DNN has been used, which is one of the most technical in the state of the art in recognition of speech. In speech processing, Deep Neural Networks (DNN) have shown to be able to learn automatically a representation of the speech information. In this project, it is represented by the Bottleneck features that are used for training the same base system, extracted from a DNN implemented in Keras (Python library) with the purpose of replacing the traditional features of conventional approaches (MFCCs). Keras is one of the most powerful libraries for evaluating and developing deep learning models.

In order to evaluate the performance of the system, different experiments have been carried out whose results and inferred conclusions are reflected in this project. The results obtained from the system based on MFCCs features and the system based on Bottleneck features have been analyzed. Different techniques such as transformations of the space of features or adaptation to the speaker have allowed to improve the performance of the initial system, being the MMI the technique that provides better Word Error Rates (WER). In addition, the hybrid HMM-DNN system, that replaces the GMMs in traditional systems, has allowed to improve the initial results even more. In this experimental scenario, the study shows that the Bottleneck features allow a performance as good as the original MFCCs but the recognition rates don't improve significantly respect the system based on MFCCs.

Keywords

Automatic Speech Recognition, Deep Neural Networks, Bottleneck, Hidden Markov Models, Kaldi, Keras.

Agradecimientos

Durante estos dos años de Máster, he intentado dar lo mejor de mí para finalizar mi etapa de estudiante. Fue un año duro; nueva ciudad, nueva universidad y nueva vida. No fue un camino fácil pero siempre conté con el apoyo de las personas más importantes.

En primer lugar, agradecer a mi familia. Ellos siempre están ahí, nunca fallan. A mi padre y a mi madre por confiar en mí y apoyarme y a mis hermanos que siempre se sintieron orgullosos de mí. Agradecer también a mis primas y tías, viajar con ellas me ayudó a evadirme cuando más lo necesitaba.

Adicionalmente, he tenido la suerte de contar con grandes amigas que vivieron mis agobios pero también mis alegrías. Clau y Patri además de compañeras de piso son amigas de la infancia. Volver a cruzarnos ha sido como volver a nuestros mejores años y contar con ellas en el día a día lo hizo todo más fácil. Y no me olvido de mi amiga María que, a pesar de la distancia, siempre está cerquita de mí.

Mil gracias a mi tutora, Alicia. Ella me guió en la realización de este Trabajo Fin de Máster y todo esto no habría sido posible sin su apoyo y paciencia. Gracias por incentivarme a profundizar en el mundo del reconocimiento de voz.

Por último, agradecer a aquellos compañeros que me he cruzado en este camino y que, a pesar de la presión y agobio, siempre mantenían una sonrisa en la cara.

Índice general

1. Introducción **1**

1.1. Objetivos . . . 2

1.2. Motivación . . . 2

1.3. Estructura de la memoria . . . 3

2. Estado del arte **5**

2.1. Reconocimiento de voz . . . 5

2.1.1. Revisión histórica . . . 5

2.1.2. Fundamento teórico . . . 7

2.1.2.1. Arquitectura de un sistema ASR . . . 7

2.1.2.2. Medidas de evaluación . . . 11

2.2. Redes Neuronales Artificiales . . . 12

2.2.1. Revisión histórica . . . 12

2.2.2. Fundamento teórico . . . 13

2.2.2.1. Conceptos básicos . . . 13

2.2.2.2. Topología Bottleneck . . . 14

3. Diseño y desarrollo experimental **17**

3.1. Base de datos . . . 17

3.2. Herramientas utilizadas . . . 18

3.2.1. Recursos software . . . 18

3.2.2. Recursos hardware . . . 19

3.3. Descripción experimental . . . 19

3.3.1. Generación del sistema base . . . 19

3.3.1.1. Preparación del diccionario . . . 20

3.3.1.2. Construcción del modelo de lenguaje . . . 20

3.3.1.3. Extracción de parámetros acústicos . . . 22

3.3.1.4. Segmentación de la base de datos . . . 23

3.3.1.5. Entrenamiento de los modelos fonéticos . . . 23

3.3.1.6. Decodificación . . . 25

3.3.2. Implementación DNNs . . . 25

4. Análisis de resultados **29**

4.1. Sistema basado en MFCC 29

4.1.1. Modelo basado en monofonemas y trifonemas ($\Delta + \Delta\Delta$) 29

4.1.2. Modelo basado en LDA+MLLT 30

4.1.3. Modelo basado en SAT 30

4.1.4. Modelo basado en MMI y fMMI 31

4.1.5. Sistema HMM-DNN (híbrido) 32

4.1.6. Análisis comparativo - Sistemas basados en MFCCS 33

4.2. Sistema basado en características Bottleneck 33

5. Conclusiones y líneas futuras **39**

Glosario de acrónimos **42**

Bibliografía **46**

Índice de figuras

2.1. Modelo conceptual de los procesos de producción y reconocimiento de voz [29]. . 7

2.2. Diagrama de bloques de un sistema global de reconocimiento de voz [29]. 8

2.3. Etapas para obtener los Coeficientes Cepstrales en las Frecuencias de Mel (MFCC) [23]. 9

2.4. HMM basado en palabras con 5 estados (de izquierda a derecha). Basada en [29]. 10

2.5. Arquitectura de una red neuronal feed-forward [21]. 13

3.1. Ejemplo del formato del archivo lexicon.txt 20

3.2. Ejemplo del formato del archivo text. 20

3.3. Ejemplo del formato del archivo segments. 21

3.4. Ejemplo del formato del archivo utt2spk. 21

3.5. Ejemplo del formato del archivo words.txt 21

3.6. Diagrama de bloques de los entrenamientos de los modelos foneticos. 24

3.7. Arquitectura del sistema híbrido DNN-HMM [16]. 25

3.8. Arquitectura de la red neuronal empleada en los experimentos - Topología Bottleneck [22]. 26

4.1. Representación gráfica de la tasa de error de palabra obtenida para cada uno de los entrenamientos del sistema basado en características MFCC. 33

4.2. Representación gráfica de la función sigmoide. 34

4.3. Representación gráfica de las tasas de error de palabra del sistema basado en en MFCCs y el sistema basado en características Bottleneck. 36

4.4. Representación gráfica de la tasa de error de palabra del sistema basado en características Bottleneck eliminando los coeficientes delta. 37

Índice de tablas

3.1. Características de la base de datos Switchboard. 18

4.1. Word Error Rate resultante del modelo basado en monofonemas y trifonemas ($\Delta + \Delta\Delta$). 29

4.2. Word Error Rate resultante del modelo basado en LDA+MLLT. 30

4.3. Word Error Rate resultante del modelo basado en SAT y SI. 31

4.4. Word Error Rate resultante del modelo basado en MMI y fMMI. 31

4.5. Parámetros de la red neuronal implementada en Kaldi. 32

4.6. Word Error Rate resultante utilizando un sistema híbrido (HMM-DNN) empleando MFCCs. 32

4.7. Parámetros de la red neuronal implementada en Keras. 35

4.8. Comparación tasas de error de palabra entre el sistema basado en en MFCCs y el sistema basado en características Bottleneck. 35

4.9. Tasas de error de palabra del sistema basado en característica Bottleneck eliminando los coeficientes delta. 36

4.10. Word Error Rate resultante utilizando un sistema híbrido (HMM-DNN) empleando características Bottleneck. 37

1

Introducción

Permitir que los ordenadores modelen nuestro mundo lo suficientemente bien como para considerar que tienen "inteligencia" ha sido el foco de más de medio siglo de investigación. Para lograr esto, una gran cantidad de información debe ser almacenada y explotada por los sistemas. Así, muchos investigadores han recurrido a los algoritmos de aprendizaje para capturar gran parte de esa información. Se ha avanzado mucho en la mejora de dichos algoritmos de aprendizaje pero el desafío de la Inteligencia Artificial (Artificial Intelligence, AI) aún permanece. Una de las ramas más destacadas de la inteligencia artificial es el aprendizaje automático (Machine Learning, ML) que proporciona a los ordenadores la capacidad de aprender sin ser explícitamente programados, es decir, no es necesario que el programador indique de manera explícita las reglas que debe seguir para lograr su tarea sino que las aprende automáticamente a partir de los datos. A su vez, el aprendizaje profundo (Deep Learning, DL) ha surgido como una nueva área de investigación del aprendizaje automático. Este enfoque propone modelar abstracciones de alto nivel de los datos empleando para ello arquitecturas compuestas por un elevado número de capas ocultas. Como un gran avance en inteligencia artificial, el aprendizaje profundo ha logrado un éxito impresionante en la resolución de grandes desafíos en muchos campos, incluido el reconocimiento automático de voz (Automatic Speech Recognition, ASR), mostrando una gran mejora en la precisión y la robustez con respecto a los enfoques convencionales, especialmente cuando se cuenta con un gran número de datos.

La tarea de reconocimiento de voz consiste en obtener de forma automática la transcripción de un segmento de voz determinado. Tradicionalmente, los sistemas de reconocimiento de voz se han basado en una parametrización clásica de la señal basada en los Coeficientes Cepstrales en las Frecuencias de Mel (Mel Frequency Cepstral Coefficients, MFCC). Con el auge de las redes neuronales profundas (Deep Neural Network, DNN) en numerosos campos, han tomado protagonismo los sistemas de reconocimiento de voz basados en características Bottleneck. Este sistema se basa en una red neuronal profunda entrenada para clasificación de unidades fonéticas y se emplea en tareas de reconocimiento automático de voz, reconocimiento de idioma y reconocimiento de locutor, entre otras. El principal objetivo de la capa Bottleneck es comprimir la información de una de sus capas para poder representar la información aprendida por las capas anteriores y poder ser utilizada como una nueva parametrización de la señal de voz a nivel de frame [22].

En este Trabajo Fin de Máster se ha diseñado y desarrollado un sistema de reconocimiento de voz basado en características acústicas (MFCC) y, una vez obtenido el sistema base, se procede a compararlo con un sistema basado en características Bottleneck y se evaluará usar estas características como entrada al sistema de reconocimiento de voz.

1.1. Objetivos

El objetivo principal para la realización de este proyecto es estudiar la utilización de características Bottleneck extraídas de una red neuronal profunda en reconocimiento del habla. Se analiza si las DNNs son más robustas y si proporcionan mejores tasas de reconocimiento que un sistema basado en coeficientes MFCC en nuestro entorno experimental en concreto, además de analizar distintas técnicas para optimizar el comportamiento del sistema. Para alcanzar dicho objetivo global se marcaron los siguientes objetivos específicos.

- Desarrollo de un sistema de reconocimiento de voz basado en características acústicas (MFCC) utilizando el software Kaldi, uno de los más utilizados en este campo, para establecer un sistema base (baseline).
- Implementación de una red neuronal profunda en Keras (Python) y extracción de características Bottleneck.
- Comparación de los resultados obtenidos a partir de los coeficientes MFCC en un sistema de reconocimiento de voz con los resultados de este mismo sistema pero que emplea como entrada las características Bottleneck.

1.2. Motivación

Según el reportaje *"El cerebro artificial que piensa por ti"* [4] publicado por el País, "El mundo se ha embarcado en un viaje a la inteligencia artificial. Los aparatos que la incorporan se han ido colando silenciosamente en nuestras vidas." Este informe afirma que la inteligencia artificial y, más concretamente, las redes neuronales están teniendo en los últimos años un gran desarrollo e impacto en diversas áreas del conocimiento. Así es que en la última década, las técnicas de deep learning han revolucionado el campo de la inteligencia artificial, aportando resultados muy superiores a los obtenidos hasta el momento con otras técnicas de aprendizaje automático. El éxito de las redes neuronales profundas en distintos campos de investigación llamó la atención de los investigadores de procesamiento de señales del habla. Adicionalmente, el habla se está volviendo cada vez más popular e interactuamos más con todo tipo de dispositivos.

Por consiguiente, la elección de este proyecto responde a varias circunstancias. La primera de ellas es el interés en profundizar en reconocimiento de voz como continuación de la asignatura Tecnologías del Habla. Se trata de un proyecto interesante que permite poner en práctica los conocimientos adquiridos. Además, tanto el reconocimiento de voz como las DNNs son temas trascendentes que tienen una gran repercusión en la sociedad actual y permiten adquirir conocimientos que abren nuevas oportunidades con vistas a una futura incursión laboral.

1.3. Estructura de la memoria

En esta sección se enumeran los distintos capítulos que conforman el proyecto realizando una breve descripción de cada uno de ellos. Más concretamente, la memoria de este Trabajo Fin de Máster está formada por 5 capítulos.

- **Capítulo 1. Introducción.** En este capítulo se definen los objetivos propuestos para la realización del proyecto, las motivaciones que incentivaron la elección del mismo y la organización de este documento.
- **Capítulo 2. Estado del arte.** Este capítulo recoge una evolución histórica de las redes neuronales profundas y del reconocimiento de voz además de los conocimientos teóricos necesarios para llevar a cabo este trabajo.
- **Capítulo 3. Diseño y desarrollo experimental.** En este capítulo se explica el procedimiento experimental para desarrollar el sistema de reconocimiento de voz. Se describe la base de datos y las herramientas empleadas, la generación del sistema base y la modificación mediante redes neuronales.
- **Capítulo 4. Análisis de resultados.** Este capítulo expone un análisis de los resultados obtenidos a partir de los coeficientes MFCC y los resultados obtenidos empleando las características Bottleneck en el sistema de reconocimiento de voz.
- **Capítulo 5. Conclusiones y líneas futuras.** El último capítulo refleja el análisis global del procedimiento experimental que se ha llevado a cabo evaluando los distintos métodos experimentales y valorando si se han cumplido los objetivos propuestos. Adicionalmente, se proponen líneas de investigación futuras.

2

Estado del arte

Se dedica el presente capítulo a describir la evolución histórica del reconocimiento automático de voz y de las redes neuronales profundas además de presentar los conocimientos teóricos necesarios para comprender el desarrollo experimental realizado.

2.1. Reconocimiento de voz

2.1.1. Revisión histórica

La voz es la manera más natural y fácil que tenemos los humanos de comunicarnos y transmitir información. Sin esta habilidad para comunicar fluidamente ideas, la sociedad actual probablemente no se habría desarrollado o al menos, no al mismo nivel. De aquí nace el interés por crear instrumentos capaces de producir y reconocer voz, imitándonos a nosotros mismos. Aunque desde el punto de vista humano es difícil de creer, el reconocimiento de voz es una tarea extremadamente compleja y es un campo de investigación muy activo desde 1950. Se ha experimentado un gran avance, partiendo de un simple y limitado reconocedor de una sola palabra a un sistema mucho más complejo, pero también más práctico, que puede reconocer el habla espontánea continua [18].

La tecnología de los sistemas de reconocimiento de voz ha progresado en cada década. El primer gran avance en la historia de ASR ocurrió en la década de los 50 cuando en los Laboratorios Bell se construyó el primer sistema de reconocimiento de voz [8][18]. Crear estos sistemas fue posible gracias a los pioneros del habla, Harvey Fletcher y Homer Dudley, quienes entendieron la importancia del espectro de frecuencias y la naturaleza fonética del sonido del habla. Los algoritmos y métodos modernos relacionados con el habla utilizan su conocimiento y procesan las señales de voz en el dominio de la frecuencia, ya que cada componente se puede modelar y procesar de forma independiente.

En la década de los 60, los sistemas ASR eran aún sistemas con un vocabulario reducido (10 a 100 palabras) y solo podían reconocer palabras aisladas. Estaban basados en propiedades fonético-acústicas del sonido, usaban análisis de banco de filtros y métodos simples de normaliza-

ción de tiempo. Sin embargo, surgió un cambio de mentalidad y se focalizó en el reconocimiento de fonemas lo que permitió abrir puertas para el reconocimiento de voz continua.

En la década de los 70, los sistemas ASR reconocían palabras conectadas y dígitos, utilizaban reconocimiento de patrones y el vocabulario aumentó a un tamaño de 100 a 1000 palabras. Previamente solo las señales de voz puras eran usadas como entradas y uno de los avances más importantes ocurrió en esta década. La introducción de los coeficientes LPC (Linear Prediction Coding), cambió completamente el enfoque de las señales de entrada. En las siguientes décadas se extraían las características de la señal de voz antes de introducirlas al sistema y se convirtió en una técnica de vanguardia [3].

Los problemas en reconocimiento de voz surgen cuando el vocabulario se amplía. Cuando el número de palabras aumenta, es más probable que las palabras usadas sean acústicamente similares y sea más fácil que se mezclen entre sí si solo se emplea la información acústica en el reconocimiento. Los modelos de lenguaje definen qué palabras pueden ser reconocidas por el sistema, lo que ayuda a restringir el espacio de búsqueda y hace que el proceso de reconocimiento sea más eficiente, ya que el sistema puede ignorar secuencias que no están permitidas en el idioma. Ahora, el sistema ASR no solo tiene que depender de la información acústica, sino que también puede usar el conocimiento de la estructura de un lenguaje para ayudar a reconocer correctamente las palabras acústicamente similares. Al principio, se utilizaron simples redes de palabras y se definieron qué palabras y secuencias de palabras estaban permitidas. El modelado del lenguaje en redes de palabras funciona adecuadamente si el vocabulario del sistema es reducido y no es necesario un reconocimiento complejo. Sin embargo, las redes de palabras no permiten ninguna secuencia fuera de las redes definidas, lo que hace que sea poco práctica en sistemas más grandes [18].

En 1970, IBM introduce modelos de lenguaje que utilizaban métodos probabilísticos. Su modelo de lenguaje utilizaba modelos de N-gramas que permitía modelar la gramática del lenguaje dando la probabilidad de que ocurra una secuencia de N palabras. Básicamente, el modelo puede hacer una predicción estadística del próximo elemento de cierta secuencia de elementos sucedida, es decir, ver N palabras en el pasado y calcular la probabilidad de cuál sería la palabra más probable para seguir a estas N palabras. Los modelos de N-gramas también permiten secuencias de palabras que no ocurrieron en el conjunto de entrenamiento, haciéndolo más flexible incluso en los entornos complejos. Se trata de una poderosa representación estadística de la gramática y se usa hoy en día en los reconocedores.

A pesar de los avances de la década de los 70, no fue posible crear un sistema de Reconocimiento de Voz Continuo de Vocabulario Grande (Large Vocabulary Continuous Speech Recognition, LVCSR) independiente del hablante con la tecnología predominante. El modelado acústico basado en plantillas se convirtió en un inconveniente, ya que las plantillas no pueden modelar una variedad natural del habla de manera eficiente. Los investigadores comenzaron a ver el muro con los métodos intuitivos y se interesaron en el potencial de los métodos estadísticos y, finalmente, en los años 80, los Modelos Ocultos de Markov (Hidden Markov Models, HMM) experimentaron un gran avance. En lugar de un patrón de plantilla, el método de modelado estocástico modela un fonema con una distribución de probabilidad, generalmente correspondiente a una mezcla de componentes gaussianas (Gaussian Mixture Model, GMM). De esta forma se puede modelar la variabilidad acústica. Los sistemas pudieron aumentar su vocabulario, lo que permitió que los sistemas ASR se usaran en un amplio rango de aplicaciones [3][18].

Uno de los mayores retos en reconocimiento de voz es la variabilidad. Cuando una persona dice una misma palabra, las señales acústicas no son idénticas e incluso pueden ser notablemente diferentes. La extracción de características es un proceso que convierte la señal de audio de entrada en vectores acústicos de tamaño fijo. Uno de los métodos más simples y más utilizados son los coeficientes MFCC que se introdujeron en la década de los 80. Este método utiliza el conocimiento de la audición humana y le da más énfasis a las frecuencias que son más importantes para que los humanos entiendan el habla.

En la década de los 90, Bourlard y Morgan probaron si las redes neuronales (Neural Network, NN) podían ser la siguiente técnica de vanguardia en el reconocimiento de voz. Construyeron un reconocedor donde se usaron redes neuronales con una sola capa oculta no lineal para reemplazar GMM en HMM. El mayor obstáculo del momento era el hardware inadecuado y los algoritmos de aprendizaje que no eran capaces de entrenar a redes neuronales con muchas capas ocultas y una gran capa de salida con una gran cantidad de datos de entrenamiento [8].

Durante la última década, los problemas con el hardware y los datos se han superado y han permitido la introducción de las redes neuronales profundas. Combinado con los nuevos métodos de entrenamiento, las DNNs (Deep Neural Networks) se han convertido en una prometedora herramienta capaz de mejorar los rendimientos de los métodos más antiguos. Hoy en día, las DNNs pueden superar a los GMM en el modelado acústico especialmente con grandes conjuntos de datos y vocabulario. Son capaces de aproximar cualquier función con la precisión deseada si se tienen suficientes capas y nodos, de manera similar a los GMM. Las redes neuronales profundas son más flexibles que los GMM y en el campo del reconocimiento de voz, se emplean también en la extracción de características y en la construcción del sistema ASR [18].

2.1.2. Fundamento teórico

2.1.2.1. Arquitectura de un sistema ASR

Un sistema de reconocimiento de voz (Automatic Speech Recognition, ASR) lleva a cabo el proceso de convertir palabras habladas a palabras escritas. Es decir, obtener de forma automática la transcripción de un segmento de voz determinado. Un sistema ASR clásico está formado por tres fuentes principales de conocimiento: un modelo acústico, un léxico fonético y un modelo lingüístico. El modelo acústico caracteriza los sonidos del lenguaje, principalmente los fonemas y sonidos adicionales (pausas, respiración, ruido de fondo, etc.); el léxico fonético contiene las palabras que el sistema puede reconocer con sus posibles pronunciaciones; y el modelo de lenguaje proporciona conocimiento sobre las secuencias de palabras que se pueden producir en ese idioma. En la Figura 2.1 se presenta un modelo conceptual simple de los procesos de generación de voz y reconocimiento de voz.

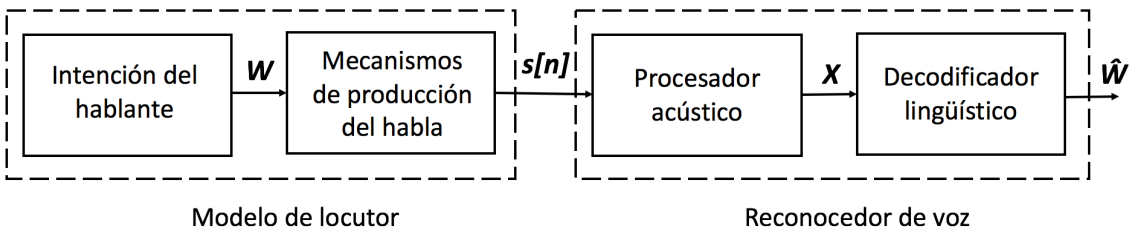

Figura 2.1: Modelo conceptual de los procesos de producción y reconocimiento de voz [29].

El hablante expresa un pensamiento como parte de una conversación con otro humano o con una máquina. Para expresar ese pensamiento, el locutor debe componer una oración lingüísticamente significativa, W, en forma de una secuencia de palabras. Una vez que se seleccionan las palabras, el hablante envía señales de control apropiadas a los órganos articulares del habla que forman una expresión cuyos sonidos son los necesarios para pronunciar la oración deseada, lo que da como resultado la forma de onda del habla s[n]. El proceso de crear la forma de onda refleja el acento del locutor y la elección de las palabras para expresar un pensamiento o una solicitud determinada. Los pasos de procesamiento del reconocedor de voz se muestran en la parte derecha de la Figura 2.1 y consisten en un procesador acústico que analiza la señal de voz y la convierte en un conjunto de características acústicas (espectrales, temporales), X, que caracterizan de manera eficiente la voz, seguidos de un proceso de decodificación lingüística que hace una estimación (máxima verosimilitud) de las palabras de la oración hablada, lo que resulta en la oración reconocida $\hat{W}$. La Figura 2.2 muestra un diagrama de bloques más detallado del sistema general de reconocimiento de voz.

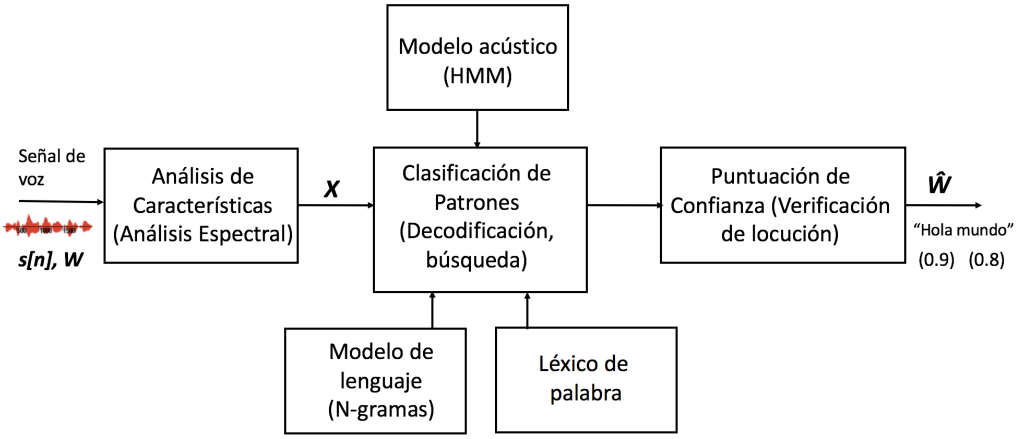

Figura 2.2: Diagrama de bloques de un sistema global de reconocimiento de voz [29].

La señal de entrada de voz, s[n], se convierte a la secuencia de vectores de características, $X = x_1, x_2, ..., x_T$, mediante el bloque de análisis de características (también denominado análisis espectral). Los vectores de características se calculan *trama a trama* (es decir, en segmentos cortos realizando un análisis localizado). En particular, los Coeficientes Cepstrales en las Frecuencias de Mel, que representan la amplitud del espectro del habla de manera compacta, son ampliamente utilizados para representar las características espectrales a corto plazo. En este Trabajo Fin de Máster se utilizan dichas características acústicas para establecer el sistema base de reconocimiento de voz. En la Figura 2.3 se muestra el proceso típico para la elaboración de un vector de características MFCC.

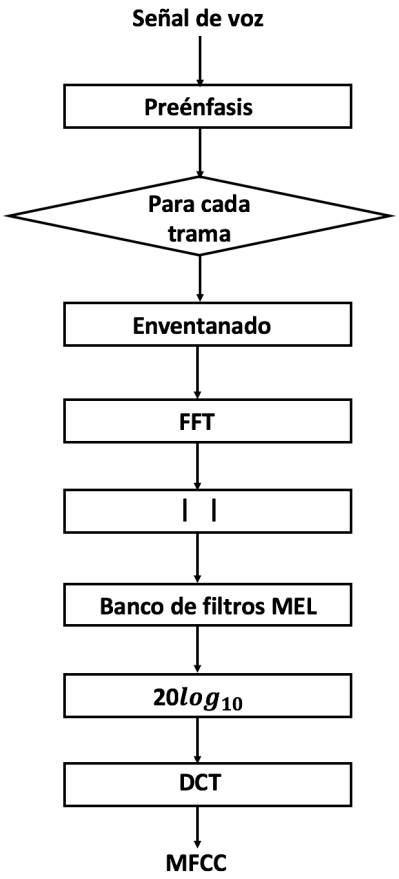

Figura 2.3: Etapas para obtener los Coeficientes Cepstrales en las Frecuencias de Mel (MFCC) [23].

En primer lugar, la señal a parametrizar se pasa por un filtro de preénfasis para compensar la atenuación de aproximadamente 20 dB/década que se produce en el proceso fisiológico del mecanismo de producción del habla. La señal de voz es un proceso aleatorio y no estacionario y esto supone un inconveniente a la hora de analizarla. No obstante, es posible salvar este problema si se tiene en cuenta que a corto plazo (en el orden de los ms) la señal es cuasi-estacionaria. Este hecho da lugar a un tipo de análisis donde se obtienen segmentos o tramas de la señal de pocos milisegundos denominado análisis localizado. A este proceso donde se generan tramas o segmentos consecutivos de señal se le denomina enventanado. Generalmente se suelen trabajar con ventanas de tipo Hamming y Hanning, con un tamaño de 20 ms. Para mantener la continuidad de la información de la señal, es muy común realizar el enventanado con bloques de muestras solapados entre sí, de tal manera que no se pierde información. Posteriormente, se calcula la Transformada Discreta de Fourier (DFT) de cada trama y a partir de este momento se descarta la fase y se trabaja con el módulo de la envolvente de la señal de voz. Se multiplica la envolvente de la señal por un banco de filtros triangulares. Estos triangulos están espaciados de acuerdo con la escala de frecuencias de Mel. Una vez que la envolvente del espectro de la señal de voz es multiplicada por el banco, se calcula la energía correspondiente en cada uno de los filtros y, tras obtener la energía, se pasa al dominio de la potencia espectral logarítmica. Para eliminar la dependencia o correlación estadística, se hace uso de la Transformada Discreta del Coseno (DCT) y, de este vector obtenido, se toman la cantidad de coeficientes deseados por trama. Adicionalmente, una vez calculados los vectores de características MFCC correspondientes a todas las tramas, es habitual aplicar una normalización de media y varianza (Cepstral Mean and Variance Normalization, CMVN) a cada dimensión (coeficiente), con el objetivo de mitigar los efectos producidos por distintas condiciones: ruido ambiente, variaciones del canal, variabilidad

interlocutor, etc.

Para incluir más información del contexto se suele introducir la velocidad y aceleración de los parámetros. Así surgen los MFCC-Delta (o ΔMFCC) y los MFCC Delta-Delta (o ΔΔMFCC), que representan la evolución temporal de los fonemas en su transición a otros fonemas. Los ΔMFCC se calculan como la variación de los coeficientes MFCC con respecto a un instante de tiempo y son denominados coeficientes de velocidad o de primera derivada. Los coeficientes ΔΔMFCC representan la variación de los coeficientes de velocidad, por lo que son llamados coeficientes de aceleración.

Si volvemos a la Figura 2.2, el bloque de clasificación de patrones decodifica la secuencia de vectores de características en una representación simbólica que es la cadena de máxima probabilidad, $\hat{W}$, que podría haber producido la secuencia de entrada de los vectores de características. El sistema de reconocimiento de patrones utiliza un conjunto de modelos acústicos y un léxico de palabras para proporcionar la puntuación de coincidencia acústica para cada cadena propuesta. Además, se utiliza un modelo de lenguaje de N-gramas para calcular una puntuación del modelo de lenguaje para cada cadena de palabras propuesta. Finalmente, el bloque final es un proceso de puntuación que se utiliza para proporcionar una puntuación de confianza para cada palabra individual en la cadena reconocida [29].

En los sistemas tradicionales de ASR, el método más utilizado para construir modelos acústicos son los Modelos Ocultos de Markov (HMM). La Figura 2.4 muestra un HMM simple de 5 estados para modelar una palabra completa. Cada estado HMM se caracteriza por una distribución (normalmente se suele elegir un GMM pero en el caso de los sistemas híbridos, sería una DNN) que caracteriza el comportamiento estadístico de los vectores de características dentro de los estados del modelo. Adicionalmente, el HMM también se caracteriza por un conjunto explícito de transiciones de estado, a_{ij}, que especifican la probabilidad de realizar una transición del estado i al estado j en cada trama, definiendo así la secuencia temporal de los vectores de características durante la duración de la palabra. Por lo general, las auto-transiciones, a_{ii} son altas (cerca de 1.0), y las transiciones de salto, a_{12}, a_{23},a_{34}, a_{45}, en el modelo, son pequeñas (cerca de 0) [29].

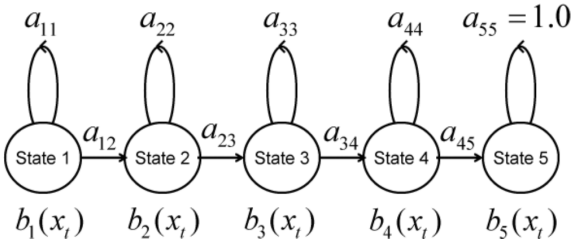

Figura 2.4: HMM basado en palabras con 5 estados (de izquierda a derecha). Basada en [29].

Con el fin de entrenar el HMM (es decir, aprender los parámetros del modelo óptimo) se utiliza el algoritmo de Baum-Welch. Este algoritmo alinea cada una de las diversas palabras (o unidades de sub-palabras) con las entradas habladas y posteriormente estima las medias, covarianzas y ganancias para las distribuciones en cada estado del modelo. El método Baum-Welch es un algoritmo que se itera hasta que se obtiene un alineamiento estable de los parámetros y el habla. Se utiliza un modelo HMM inicial para comenzar el proceso de entrenamiento. El modelo inicial puede elegirse al azar o seleccionarse en función de un conocimiento *a priori* de

los parámetros del modelo. El bucle de iteración es un procedimiento de actualización simple para calcular las probabilidades del modelo hacia adelante y hacia atrás basándose en una base de datos de voz de entrada y luego optimizar los parámetros del modelo para proporcionar un HMM actualizado. Este proceso se itera hasta que no se produce ninguna mejora adicional en las probabilidades con cada nueva iteración.

2.1.2.2. Medidas de evaluación

El tipo de error que comete un sistema de reconocimiento de voz se determina comparando el resultado obtenido con la transcripción correcta. Se pueden distinguir tres tipos de errores:

- Sustitución: en la posición de una unidad (palabra o fonema) se reconoce una unidad diferente.
- Eliminación: en el resultado de reconocimiento se omite una unidad.
- Inserción: el resultado contiene unidades adicionales que no son habladas.

Las medidas más comunes para evaluar el rendimiento de un sistema de reconocimiento de voz se pueden calcular a nivel de fonema o palabra y son la corrección, la precisión y el error. La corrección se define con la siguiente expresión:

$$Corr = \frac{N - D - S}{N}$$

siendo N el número de unidades en la transcripción correcta, D el número de eliminaciones y S el número de sustituciones. La precisión es similar a la expresión de corrección, pero tiene en cuenta el número de inserciones I:

$$Acc = \frac{N - D - S - I}{N}$$

Ambas medidas se dan comúnmente en notación porcentual. Finalmente, la tasa de error es la suma de todos los tipos de errores entre el número de unidades en la transcripción y está estrechamente relacionada con la precisión.

$$Err = \frac{D + S + I}{N} = 100\,\% - Acc$$

Si las unidades son palabras, la tasa de error se denomina *Word Error Rate* (WER), que representa la tasa de error de palabra. Esta tasa es la que se utiliza típicamente y es la que se va a calcular en los experimentos que se han llevado a cabo en este Trabajo Fin de Master.

2.2. Redes Neuronales Artificiales

2.2.1. Revisión histórica

Diseñar y construir máquinas capaces de realizar procesos con cierta inteligencia ha sido uno de los principales objetivos de los científicos a lo largo de la historia. Las redes neuronales empezaron a tener una mayor relevancia a partir de la década 1980 pero los primeros estudios y avances tuvieron lugar años atrás. Seguidamente, se describe una evolución histórica que abarca desde la década 1930 hasta la actualidad.

El primer paso fue dado en el año 1936 por Alan Turing con el estudio del cerebro como una forma de ver la computación. Escribió el artículo *«Computing machinery and intelligence»* llamado a revolucionar la informática y a crear un nuevo campo de investigación: la inteligencia artificial. Sin embargo, fueron Warren S. McCulloch y Walter Pitts en el año 1943 quienes concibieron los fundamentos de la computación neuronal lanzando una teoría sobre la forma de trabajar de las neuronas y modelando una red neuronal simple mediante circuitos eléctricos. El modelo neuronal ha sido tomado como punto de partida para el desarrollo de muchos de los modelos neuronales actuales y, además, es utilizado como punto de referencia para evaluar el comportamiento de otros modelos [28].

En 1949, Donald Hebb formó las bases de la teoría de las redes neuronales. Explicó los procesos del aprendizaje desde un punto de vista psicológico defendiendo la siguiente idea: las conexiones sinápticas se fortalecen cuando dos o más neuronas se activan de forma contigua en el tiempo y en el espacio. La propuesta de Hebb tuvo un fuerte impacto, llegando a constituir el núcleo de muchos planteamientos desarrollados en las décadas posteriores y sigue siendo una referencia muy importante en este campo en la actualidad [32].

En la década 1950 fueron numerosos los avances. Karl Lashley estudió los fenomenos de aprendizaje y la discriminación sensorial y destacó que el proceso de aprendizaje es un proceso distribuido y no local a una determinada área del cerebro. En el año 1956, se considera el nacimiento de la inteligencia artificial con el Congreso de Dartmouth en el que se discutió acerca de las máquinas y la posibilidad de comportarse de manera inteligente. Adicionalmente, Frank Rosenblatt presenta en el año 1957 una nueva aproximación al reconocimiento de patrones mediante la introducción del perceptrón, la red neuronal más antigua que permite reconocer una serie de patrones después de aprender otros similares aunque no se hayan empleado en el entrenamiento. Introdujo los principios de la Neurodinámica y confirmó que el aprendizaje del perceptrón convergía hacia un estado finito (Teorema de Convergencia del Perceptrón).

Se desarrolló la primera red neuronal aplicada a un problema real (filtros adaptativos para eliminar ecos en las líneas telefónicas) en el año 1960 y se siguió empleando durante varias décadas. Se trata del modelo Adaline (ADAptative LINear Elements) que fue desarrollado por Bernard Widroff y Maccian Hoff y se trata de un sistema adaptativo que puede aprender de forma más precisa y rápida que los perceptrones existentes [32]. En el año 1969, Minsky y Papert manifestaron las limitaciones del perceptrón y, por consiguiente, se paralizó el avance de este campo de la inteligencia artificial durante 10 años y las redes neuronales pasaron a un segundo plano. Demostraron que el perceptron únicamente servía para clasificar problemas linealmente separables que ya se llevaban a cabo mediante métodos estadísticos y de una forma mucho más eficiente.

En el año 1985, resucitó el interés en las redes neuronales con el libro de John Hopfield, *"Computación neuronal de decisiones en problemas de optimización"* y en el año 1986, David Rumelhart y G. Hinton redescubrieron el algoritmo de aprendizaje de propagación hacia atrás (backpropagation). A partir de la década de los 80, el número de trabajos sobre redes neuronales ha aumentado exponencialmente debido a que se superaron los problemas con el hardware y los datos, apareciendo un gran número de aportaciones tanto a los métodos de aprendizaje como a las arquitecturas y aplicaciones de las redes neuronales, entre ellas, el reconocimiento de voz [32].

2.2.2. Fundamento teórico

2.2.2.1. Conceptos básicos

Una red neuronal artificial (Artificial Neural Network o Neural Network, NN) es un modelo de computación que se inspiró en una red neuronal real, el cerebro. Se utiliza para aproximar funciones no lineales y está compuesto por un gran número de elementos simples, elementos de procesos interconectados que procesan información por medio de su estado dinámico como respuesta a entradas externas. Las unidades de procesamiento simples que constituyen una red neuronal se denominadan neuronas. Las neuronas están interconectadas entre sí y cada conexión tiene cierto peso que describe la fortaleza de la conexión entre ellas.

La red neuronal suele estar estructurada de modo que tenga un número específico de capas. La primera capa es la capa de entrada que se alimenta con vectores de entrada que representan los datos, en las capas ocultas se aplica una transformación a la salida de la capa anterior y la capa de salida obtiene la salida de la red. Todos los cálculos tienen lugar en las capas ocultas y puede haber tantas como sea necesario, formando lo que se conoce como DNN. Un ejemplo de red neuronal se muestra en la Figura 2.5.

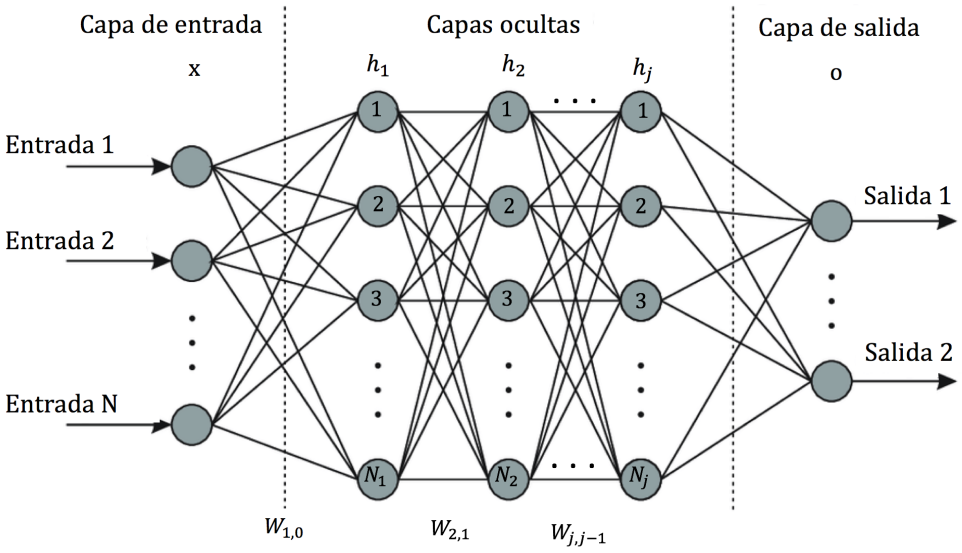

Figura 2.5: Arquitectura de una red neuronal feed-forward [21].

Cada neurona está caracterizada, en cualquier instante, por un valor numérico denominado valor o estado de activación; asociado a cada unidad, existe una función (típicamente no lineal),

que transforma el estado actual de activación en una señal de salida. Dicha señal es enviada a través de los canales de comunicación unidireccionales a otras unidades de la red, en estos canales la señal se pondera de acuerdo con el peso ($W_{j,j-1}$) asociado a cada uno de ellos [12].

Si suponemos un conjunto de entrenamiento supervisado ($x^{(i)}$, $y^{(i)}$), donde $x^{(i)}$ es un vector de características dado, e $y^{(i)}$ su clase correspondiente (valor verdadero), cada capa oculta aplica una transformación no lineal g a la salida de la capa anterior. Esta transformación tiene en cuenta las matrices de pesos $W_{j,j-1}$ y los vectores umbrales b_j que relacionan una capa con la anterior y proporciona los valores de activación de las neuronas con la siguiente ecuación [21]:

$$h_j(x^{(i)}) = g(W_{j,j-1}h_{j-1}(x^{(i)}) + b_j), \qquad j = 2, ..., N-1$$

El número de capas y neuronas ocultas se determina según el tamaño de los datos de entrenamiento; el conjunto de entrenamiento más grande generalmente permite ser modelado con redes más grandes. Cuando aumenta el número de neuronas, también aumentan los parámetros libres que deben estimarse durante el entrenamiento, por lo que se debe buscar un equilibrio entre la complejidad del modelo y la duración del entrenamiento y el uso, como en los sistemas basados en GMM-HMM.

Durante el entrenamiento de la red, se realiza el ajuste de los pesos de cada neurona de forma iterativa, hasta que los valores de salida se corresponden con los esperados, usando un conjunto de datos de entrenamiento. Es necesario definir el método de aprendizaje por el cual las conexiones varían para proporcionar la salida deseada. Existen dos tipos de aprendizaje:

1. **Aprendizaje no supervisado**. Se desconoce la señal deseada $y^{(i)}$ que debe proporcionar la red neuronal. Es la propia red la que busca patrones y características comunes entre los datos y ajusta sus pesos y umbrales según estos. Estos sistemas proporcionan un método de clasificación de las diferentes entradas mediante técnicas de agrupamiento o *clustering*.
2. **Aprendizaje supervisado**. Presenta a la red las salidas que debe proporcionar ($y^{(i)}$) antes las señales que se le presentan. Se observa la salida obtenida de la red y se determina la diferencia entre ésta y la señal deseada. Este aprendizaje es el empleado en los distintos experimentos del Trabajo Fin de Master.

2.2.2.2. Topología Bottleneck

Una red neuronal cuya capa oculta intermedia tiene un número reducido de neuronas en comparación con las otras capas, fuerza a la red a concentrar la información en un número reducido de valores. A esta topología de red neuronal se conoce como *bottleneck*. El principal objetivo de la capa bottleneck es comprimir la información de una de sus capas para poder representar la información aprendida por las capas anteriores y poder ser utilizada en este trabajo como una nueva parametrización de la señal de voz a nivel de frame.

Las DNNs con esta topología proporcionan buenos resultados en el campo de reconocimiento del habla y se ha reflejado en diversos estudios que son superiores a otros tipos de arquitecturas a la hora de extraer características para el reconocimiento [14]. Este tipo de DNN, pertenece al conjunto de algoritmos de aprendizaje supervisado que utilizan etiquetas para aprender, a

diferencia de las técnicas de aprendizaje no supervisadas, que funcionan sin ningún conocimiento previo sobre las etiquetas de los datos de entrenamiento. En el Capítulo 3, se explica más detalladamente la topología Bottleneck implementada en este Trabajo Fin de Master además de la extracción de características y el entrenamiento que se ha llevado a cabo.

3

Diseño y desarrollo experimental

El presente capítulo describe la base de datos y las herramientas software y hardware utilizadas en los experimentos además del procedimiento experimental que se ha llevado a cabo para desarrollar el sistema de reconocimiento automático de voz.

3.1. Base de datos

La base de datos es un elemento fundamental a la hora de generar un sistema de reconocimiento del habla. Su calidad y alcance afectan en gran medida las actividades de los sistemas de reconocimiento y, por consiguiente, cada base de datos acústico-fonética debe contener en sí misma una gran riqueza fonética. La base de datos seleccionada para la realización de este Trabajo Fin de Master es *Switchboard-1 Release 2*, número de serie LDC97S62 [5]. El Consorcio de Datos Lingüísticos (LDC), que controla esta base de datos, es un consorcio abierto de universidades, empresas y laboratorios de investigación que apoya la educación relacionada con el idioma, la investigación y el desarrollo de la tecnología mediante la creación y el intercambio de recursos lingüísticos, incluidos datos, herramientas y estándares.

Switchboard-1 Release 2 es una base de datos de múltiples locutores y es de interés para investigadores de reconocimiento de locutor y reconocimiento de voz. Está compuesta por conversaciones telefónicas en inglés entre dos locutores humanos que discuten sobre temas generales definidos. Se proporcionaron unos 70 temas y los locutores no podían conversar a la vez ni hablar sobre un tema determinado más de una vez. Registra el habla espontánea, con frecuentes superposiciones e interrupciones entre los hablantes y ruidos de fondo. Recopilada por Texas Instruments en 1990 con la financiación de DARPA, el conjunto completo incluye aproximadamente 260 horas de conversación, 2.400 conversaciones telefónicas con un promedio de 6 minutos de duración; en otros términos, más de 240 horas de habla grabada y aproximadamente 3 millones de palabras de texto habladas por 543 locutores (302 hombres, 241 mujeres) de todas las áreas de los Estados Unidos. La Tabla 3.1 recoge las características de la base de datos descrita.

Características	**Valor**
Nombre	Switchboard-1 Release 2
Autores	John J. Godfrey, Edward Holliman
LDC Nº Serie	LDC97S62
Año	1993, 1997
Idioma	Inglés (Americano)
Fuente de Datos	Conversaciones telefónicas
Tamaño	14610176 KB
Nº Locutores	543
Nº Conversaciones	2430
Duración promedio cada conversación [min]	6
Hombres [%]	55,62
Mujeres [%]	44,38
Tasa de muestreo [Hz]	8000
Canales de audio	2 canales u-law
Aplicaciones	Reconocimiento de voz y locutor

Tabla 3.1: Características de la base de datos Switchboard.

3.2. Herramientas utilizadas

3.2.1. Recursos software

Kaldi

Kaldi comenzó su desarrollo en 2009 en la Universidad Johns Hopkins y es un conjunto de herramientas para reconocimiento de voz escrito en C ++ y bajo la Licencia Apache v2.0. Está diseñado para ser utilizado por investigadores de reconocimiento del habla y proporciona recetas para la construcción de sistemas de reconocimiento de voz, que funcionan a partir de bases de datos disponibles, como las que proporciona el Consorcio de Datos Lingüísticos. Es un software libre y de código abierto, que está continuamente en desarrollo [20]. Se usa esta herramienta en la realización de este Trabajo Fin de Master para desarrollar el sistema base basado en características acústicas y representa actualmente una de las herramientas ASR más usadas en sistemas del estado del arte.

Keras

Keras es una API de redes neuronales de alto nivel, escrita en Python y de código abierto. Fue desarrollada por François Chollet y no maneja computación de bajo nivel. En su lugar, utiliza otra librería para hacerlo, llamada "Backend" (TensorFlow, CNTK o Theano). La API de alto nivel de Keras maneja la forma en la que se crean los modelos, definiendo capas o configurando múltiples modelos de entrada-salida. En este nivel, Keras compila el modelo con funciones de optimización y pérdida, proceso de entrenamiento con función de ajuste. Se utiliza Keras en este Trabajo Fin de Master para implementar una red neuronal profunda y extraer las características Bottleneck.

Matlab

MATLAB es un software matemático que ofrece un entorno de desarrollo integrado con un lenguaje de programación propio (lenguaje M). Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. El paquete software ha estado disponible comercialmente desde 1984 y es muy usado en universidades y centros de investigación y desarrollo [1]. Se ha utilizado esta herramienta en este Trabajo Fin de Master para representar gráficamente los resultados obtenidos y así, poder analizarlos.

Otras herramientas

Se ha empleado el sistema operativo Linux puesto que Kaldi, el software principal de este proyecto, ha sido desarrollado principalmente para él. Además, se han utilizado los lenguajes de programación Python y Bash para realizar varias de las tareas, como el tratamiento de archivos y las diferentes llamadas a las herramientas de Kaldi.

3.2.2. Recursos hardware

Implementar redes neuronales a gran escala es extremadamente costoso desde el punto de vista computacional. Las GPU aceleran considerablemente el entrenamiento de las redes neuronales con grandes cantidades de datos, lo que permite estudiar más rápidamente nuevos algoritmos y técnicas de aprendizaje. Por consiguiente, para implementar las redes neuronales en nuestros experimentos, se ha hecho uso de tarjetas gráficas o GPU para soportar la carga de trabajo.

Adicionalmente, para la realización de este proyecto se ha empleado un ordenador con un procesador Intel(R) Core(TM) i5-6500 CPU @ 3.20GHz que cuenta con una memoria RAM de 32 GB.

3.3. Descripción experimental

En esta sección se detallan los pasos seguidos para el desarrollo del sistema base de reconocimiento de voz además de la implementación de la red neuronal para extraer las características Bottleneck.

3.3.1. Generación del sistema base

Para desarrollar el sistema base o *baseline* se ha adaptado una de las recetas de Kaldi correspondiente al corpus Switchboard que proporciona Kaldi (swbd/s5c). Se actualizaron los scripts correspondientes a la receta con la finalidad de adaptarlos a nuestro entorno y realizar el entrenamiento de un sistema ASR basado en HMMs. Seguidamente, se describe el procedimiento seguido.

3.3.1.1. Preparación del diccionario

En primer lugar, se prepara un diccionario fonético de las palabras a partir de las transcripciones de los archivos (script *swbd*1*_prepare_dict.sh*) y se guarda en el fichero *lexicon.txt* el léxico de pronunciación del idioma (en este caso, en inglés) que presenta el siguiente formato (Figura 3.1).

```
!sil sil
-'s z
-'s s
-'t k uh d en t
-1k w ah n k ey
-able ax b ax l
-ains t ih n
```

Figura 3.1: Ejemplo del formato del archivo lexicon.txt

Es decir, el primer elemento corresponde con la palabra y el resto de elementos son los fonemas correspondientes. Adicionalmente, se generan ficheros con los fonemas silenciosos y no silenciosos (*nonsilence_phones.txt* y *silence_phones.txt*).

3.3.1.2. Construcción del modelo de lenguaje

A continuación, se preparan los archivos de la base de datos Switchboard-1 Release 2 con la finalidad de obtener el modelo de lenguaje del sistema de referencia. Se trata de un modelado estadístico calculado a partir de la frecuencia de las diferentes secuencias de palabras. Para ello, se obtienen los siguientes archivos (script *swbd*1*_data_prep.sh*):

- text. Este fichero contiene las transcripciones de cada una de las locuciones de los archivos de la base de datos y presenta el siguiente formato.

```
sw02005-B_045698-046049 oh yeah sure realistically it is yes
sw02005-B_046646-046800 um-hum
sw02005-B_047098-047299 that would certainly help yeah
sw02005-B_047455-047575 um-hum
sw02005-B_047899-048427 right that's just a matter of defining priorities i guess or some priorities anyway
sw02005-B_048427-048599 i think your right
sw02005-B_048948-049473 okay was good talking to you yeah take care bye-bye
sw02005-B_049473-049882 [noise]
sw02006-A_001596-001735 right
```

Figura 3.2: Ejemplo del formato del archivo text.

  Como se puede ver en la Figura 3.2, el primer elemento es el identificador de la locución (<utterance id>) y el segundo elemento es la transcripción correspondiente (<text>).

- segments. El archivo segments contiene el inicio y fin de cada locución de los archivos de audio. Este archivo contiene cuatro elementos: <utterance id>, <recording-id>, <inicio> y <fin>. El primero corresponde con la identificación de la locución, el segundo es la identificación de la grabación a la que pertenece la locución y los dos últimos corresponden al inicio y al fin de la locución en el tiempo medido en segundos.

```
sw02001-A_000098-001156 sw02001-A 0.98 11.56
sw02001-A_001980-002131 sw02001-A 19.8 21.31
sw02001-A_002736-002893 sw02001-A 27.36 28.93
sw02001-A_003390-004012 sw02001-A 33.9 40.12
sw02001-A_004012-004155 sw02001-A 40.12 41.55
sw02001-A_005013-005157 sw02001-A 50.13 51.57
sw02001-A_005696-006559 sw02001-A 56.96 65.59
sw02001-A_006960-007110 sw02001-A 69.6 71.1
sw02001-A_007332-007541 sw02001-A 73.32 75.41
sw02001-A_007642-008004 sw02001-A 76.42 80.04
sw02001-A_008143-009209 sw02001-A 81.43 92.09
```

Figura 3.3: Ejemplo del formato del archivo segments.

- utt2spk. Este fichero contiene el mapeo de cada locución a su locutor correspondiente y, por consiguiente, indica la identificación de la locución (<utterance-id>) y la identificación del locutor al que corresponde la locución (<speaker-id>).

```
sw02001-A_000098-001156 sw02001-A
sw02001-A_001980-002131 sw02001-A
sw02001-A_002736-002893 sw02001-A
sw02001-A_003390-004012 sw02001-A
sw02001-A_004012-004155 sw02001-A
sw02001-A_005013-005157 sw02001-A
sw02001-A_005696-006559 sw02001-A
sw02001-A_006960-007110 sw02001-A
sw02001-A_007332-007541 sw02001-A
sw02001-A_007642-008004 sw02001-A
sw02001-A_008143-009209 sw02001-A
```

Figura 3.4: Ejemplo del formato del archivo utt2spk.

- spk2utt. Relaciona el locutor con las locuciones que le corresponden pero, a diferencia de utt2spk, contiene en primer lugar la identificación del locutor y, seguidamente, los identificadores de las locuciones: <speaker-id> <utterance-id1> <utterance-id2>.
- wav.scp. Este archivo contiene el nombre de todos los archivos de audio y la ubicación de cada uno en el sistema.

A partir de los ficheros de los datos de entrenamiento, se construye un modelo de lenguaje inicial que consiste en una serie de ficheros que proporcionan información sobre las palabras que forman el léxico, el conjunto de fonemas, la topología del modelado e información adicional que necesita Kaldi para entrenar el modelo de lenguaje (script *prepare_lang.sh*). Estos ficheros son los siguientes:

- words.txt. Este archivo contiene todas las palabras del léxico en formatos textual y entero, con el propósito de realizar el mapeo entre los dos elementos y utilizar el formato más conveniente como argumento de entrada en las diferentes funciones ejecutables.

```
<eps> 0
!sil 1
-'s 2
-'t 3
-1k 4
-able 5
-ains 6
-an 7
-arious 8
```

Figura 3.5: Ejemplo del formato del archivo words.txt

- phones.txt. Con el mismo objetivo que el archivo words.txt, este fichero contiene los fonemas en formatos textual y entero.
- oov.txt. Este archivo incluye la palabra empleada para mapear todas las palabras fuera de diccionario (<oov>, out of vocabulary).
- topo. En este fichero se especifica la topología de los HMMs, indicando los estados y las transiciones entre ellos.
- L.fst. Contiene el léxico en formato FST (Finite State Transducer), con los fonemas como entrada y las palabras como salida.

El siguiente paso, es entrenar el modelo de lenguaje (script $swbd1\_train\_lms.sh$). El objetivo es ver cuál es la probabilidad de que una unidad lingüística vaya seguida de otra y generar la distribución de probabilidad. En este caso, se genera un modelo de tri-gramas seleccionando las palabras de las transcripciones de tres en tres.

Una vez entrenado y optimizado el modelo de lenguaje, es preciso darle otro formato para que pueda ser utilizado también a la hora de decodificar los modelos fonéticos dado que Kaldi necesita de un modelo de lenguaje en forma de transductores de estados finitos para construir los grafos de decodificación. Para ello, se emplea la herramienta SRILM que permite llevar a cabo la conversión del modelo de lenguaje del formato ARPA al formato FST (script $format\_lm\_sri.sh$). Como resultado de la conversión, los ficheros que componen el modelo de lenguaje son los mismos que los creados y, además, se genera el archivo G.fst que contiene la gramática en formato FST, donde tanto la entrada como la salida son palabras.

3.3.1.3. Extracción de parámetros acústicos

La extracción de MFCCs es el primer paso para poder tratar la señal de audio y desarrollar el sistema base. Representan el habla basados en la percepción auditiva humana y sirven de base para construir los modelos fonéticos. Se han obtenido las características acústicas de los archivos "*.wav*" con el siguiente procedimiento (script $make\_mfcc.sh$). En primer lugar, se aplica un filtro preénfasis y se divide la señal en tramas. Como se ha descrito en el Capítulo 2, a este proceso donde se generan tramas o segmentos consecutivos de señal se le denomina enventanado y, para ello, en Kaldi se emplea una ventana "Povey" con una duración de 25 ms y 10 ms de solape entre las tramas consecutivas. La ventana "Povey" es similar a la ventana Hamming pero tiende a cero suavemente en los bordes y la variación temporal se define mediante la siguiente expresión.

$$w(n) = (\frac{1}{2} - \frac{1}{2}cos(\frac{2\pi n}{N-1}))^{0,85}$$

donde $w(n)$ es el valor de la ventana y N corresponde con la longitud de dicha ventana. Seguidamente, se obtiene la energía de la señal y se calcula la Transformada Discreta de Fourier. La amplitud obtenida se pasa al dominio de Mel y se calcula el logaritmo de estas energías. Por último, se aplica la Transformada Discreta del Coseno y se toman los coeficientes deseados por cada trama, en este caso, 13 coeficientes.

Se ha realizado la técnica de normalización denominada Cepstral Mean and Variance Normalization (CMVN) que permite forzar todas las iteraciones a media 0 y varianza 1 (script

compute_cmvn_stats.sh). Se elimina de esta manera los errores introducidos por las diferencias existentes en las condiciones de los locutores o de las grabaciones (variabilidad de canal).

3.3.1.4. Segmentación de la base de datos

Se han creado distintos subconjuntos de los datos para cada uno de los experimentos (script *subset_data_dir.sh*), que consisten en un número específico de locuciones. Se han utilizado las primeras 4000 locuciones como conjunto de evaluación, que nos permite analizar el rendimiento del sistema y se ha creado un subconjunto de 100000 locuciones cortas que facilitan el alineamiento y permite agilizar el entrenamiento. Adicionalmente, se seleccionan de manera arbitraria 30000 locuciones del subconjunto anterior para entrenar el sistema basado en monofonemas. Se eliminan los excesos de expresiones si aparecen más de un número determinado de veces con la misma transcripción en un subconjunto de datos (script *remove_dup_utts.sh*). Por último, se crean dos subconjuntos para las distintas etapas del entrenamiento; uno con las primeras 100000 locuciones y el otro con el conjunto completo de entrenamiento (sin el subconjunto de evaluación).

3.3.1.5. Entrenamiento de los modelos fonéticos

El siguiente paso en el desarrollo del sistema base es el entrenamiento de los modelos fonéticos y el método empleado para construir dichos modelos acústicos son los Modelos Ocultos de Markov (HMMs). Los modelos fonéticos se realizan de forma sucesiva, dado que cada uno de ellos está basado en el anterior y la idea es la misma: alinear el modelo anterior con los datos de entrenamiento y emplear posteriormente el nuevo algoritmo de entrenamiento. Por tanto, se espera que mejoren los resultados con cada nuevo entrenamiento. Partimos de etiquetas de palabras y en el primer experimento se obtienen las etiquetas de los fonemas de los datos. Seguidamente, los resultados de estos experimentos se emplean como base para los experimentos que utilizan distintas técnicas para el etiquetado de trifonemas.

En primer lugar, se entrena un modelo de monofonemas que es un modelo acústico que no incluye ninguna información contextual sobre el fonema anterior o posterior. Se utiliza como un bloque de construcción para los modelos de trifonemas, que hacen uso de información contextual. Para ello, se utilizan los coeficientes MFCC con primera y segunda derivada (delta+delta-delta, "$\Delta + \Delta\Delta$") sobre el subconjunto de entrenamiento de 30000 locuciones cortas (entrenamiento mono). Para optimizar la estimación de los parámetros del modelo acústico se incluye una fase de alineamiento. Al alinear el audio con la transcripción de referencia utilizando el modelo acústico más actual, los algoritmos de entrenamiento pueden usar esta salida para mejorar o refinar los parámetros del modelo. Por lo tanto, cada paso de entrenamiento será seguido por un paso de alineamiento donde el audio y el texto se pueden realinear (script *align_si.sh*). El alineamiento realizado para el modelo de monofonemas corresponde con el experimento mono_ali.

A continuación, se entrena un modelo de trifonemas, en el que ya se tiene en cuenta el contexto fonético, tomando el fonema anterior y el posterior (script *train_deltas.sh*). Se utilizan delta+delta-delta sobre el subconjunto de 100000 locuciones. Se obtiene el primer entrenamiento de trifonemas, tri1 y se alinean con el experimento anterior (tri1_ali). Seguidamente, se lleva a cabo un nuevo entrenamiento, tri2, empleando también el subconjunto de 100k locuciones. Sobre su resultado se realiza un nuevo alineamiento, (tri2_ali).

El siguiente paso es aplicar sobre la base del modelo con trifonemas, las técnicas Linear Discriminant Analysis y Maximum Likelihood Linear Transform (LDA + MLLT) empleando el script *train_lda_mllt.sh*. LDA es un método de transformación lineal que puede ser usado para clasificación y reducción dimensional, toma los vectores de características y construye los estados HMM, pero con un espacio de características reducido para todos los datos. MLLT es una técnica de estimación basada en la transformación lineal de características LDA y se emplea para mejorar la toma de decisión. Se obtiene el modelo tri3 y tras los alineamientos, tri3_ali_nodup.

La siguiente técnica empleada es Speaker Adaptive Training (SAT) que realiza la normalización de los locutores y el ruido adaptándose a cada locutor específico con una transformación de datos particular (script *train_sat.sh*). Esto da como resultado datos más homogéneos o estandarizados y se realiza sobre el conjunto entero de datos de entrenamiento (entrenamiento tri4). Adicionalmente, a partir de este punto, para alinear que usa un método de regresión lineal por espacio de características de máxima verosimilitud (Maximum Likelihood Linear Regression, MLLR) y se crea el modelo tri4_ali_nodup (script *align_fmllr.sh*).

Las últimas técnicas empleadas en el entrenamiento de los modelos fonéticos son Maximum Mutual Information (MMI) y featured Maximum Mutual Information (fMMI) (scripts *train_mmi.sh* y *train_mmi_fmmi.sh*). A diferencia de los entrenamientos anteriores, se realizan varias iteraciones, intentando ajustarse a los datos y los alineamientos. La información mutua es una medida de la información proporcionada por dos variables y permite medir la dependencia mutua entre ellas. Es decir, mide la reducción de la incertidumbre de una variable aleatoria debido al conocimiento del valor de la otra. Por tanto, la técnica MMI se basa en maximizar la información mutua promedio entre el conjunto de secuencias de entrenamiento y el conjunto de modelos, y permite hacer una aproximación más eficaz. Se realiza el entrenamiento MMI sobre LDA + MLLT sobre el conjunto entero y se obtiene el modelo tri4_mmi_b0.1. Por otro lado, el algoritmo de entrenamiento discriminativo del espacio fMMI realiza una transformación lineal en los vectores de características [13]. En este caso, también se realiza el entrenamiento sobre el conjunto entero y se genera el modelo tri4_fmmi_b0.1. En la Figura 3.6, se ilustra el diagrama de bloques de los distintos entrenamientos descritos.

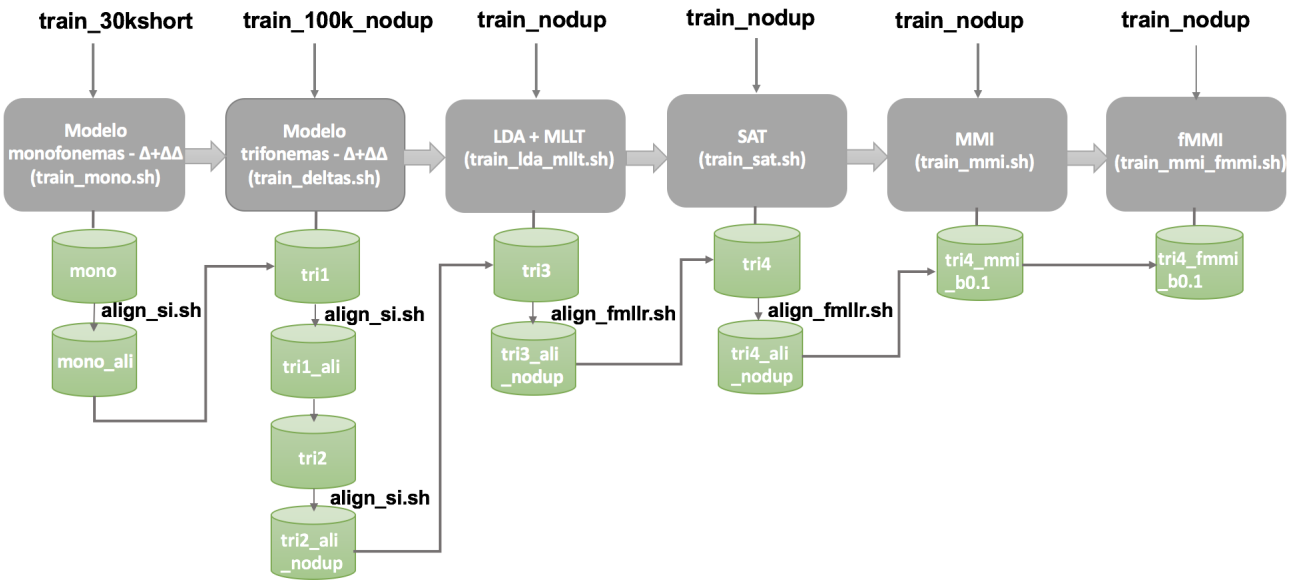

Figura 3.6: Diagrama de bloques de los entrenamientos de los modelos foneticos.

Tras las distintas transformaciones descritas, se lleva a cabo un experimento con un sistema HMM-DNN (híbrido) en el que la DNN se combina con el HMM y reemplaza al GMM. Dicha DNN se utiliza para estimar las probabilidades *a posteriori* de un estado HMM y se entrena a partir de los alineamientos generados por el sistema GMM-HMM: MFCC ($\Delta + \Delta\Delta$) + LDA + MLLT + fMLLR (características de 40 dimensiones). En la Figura 3.7 se ilustra la arquitectura de un sistema híbrido. Se han realizado experimentos con las características MFCCs y las características BN utilizando este sistema híbrido. En el Capítulo 4 se describe más en detalle el sistema DNN-HMM implementado.

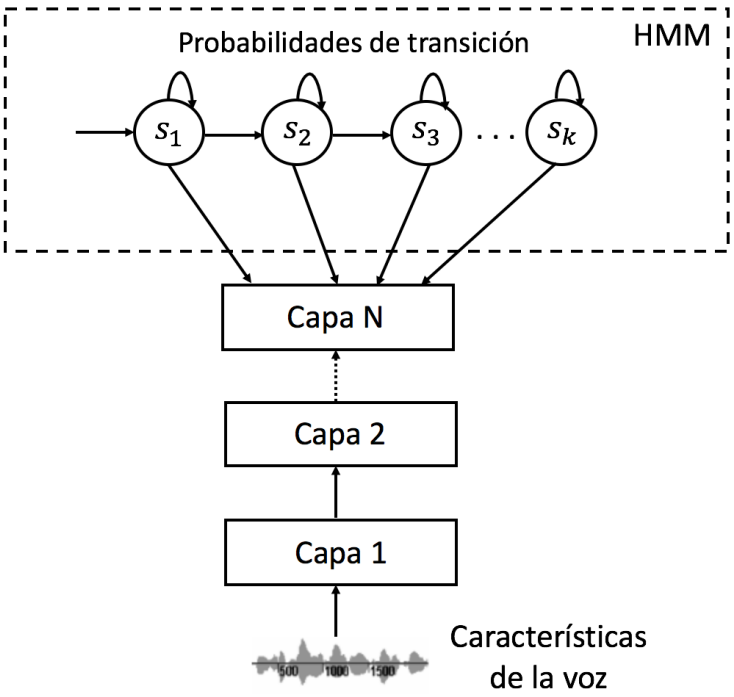

Figura 3.7: Arquitectura del sistema híbrido DNN-HMM [16].

3.3.1.6. Decodificación

Para evaluar los distintos experimentos, se realiza la decodificación (scripts *decode.sh* y *decode_fmmi.sh*) que permiten obtener las medidas WER para los seis modelos fonéticos. Se utilizan el subconjunto de evaluación (4000 locuciones) y los grafos en forma de transductor de estados finitos que recogen las características del modelo de lenguaje, la topología HMM, el diccionario fonético y la dependencia de contexto. La decodificación de los grafos se basa en el algoritmo de Viterbi y nos permite obtener las distintas transcripciones.

3.3.2. Implementación DNNs

En esta sección, se describe la implementación de la red neuronal para extraer las características Bottleneck que, posteriormente, serán utilizadas en el sistema base desarrollado. Para ello, se ha utilizado la librería Keras y se han generado los siguientes scripts. Se entrena la red neuronal con las características MFCC (*Script_DNN.py*) y, una vez la DNN está entrenada, se extrae una nueva representación con las características Bottleneck (*Script_Bott.py*).

Los archivos de entrada a la red neuronal son los vectores de etiquetas de estados de trifonemas y las matrices características MFCC que han sido obtenidas a partir de la información generada en el proceso de desarrollo del sistema base. Para obtener las etiquetas se ha utilizado la función *ali-to-pdf* de Kaldi que convierte los identificadores de transiciones en identificadores de fonema de los segmentos de audio que aparecen en los archivos de alineamiento, formando así las etiquetas de entrenamiento de la red neuronal.

La arquitectura de la red neuronal se basa en una topología Bottleneck. Está formada por una serie de capas de neuronas secuenciales: una capa de entrada, cuatro capas ocultas y una capa de salida. Una de las capas intermedias está diseñada para ser relativamente pequeña con respecto a las otras, lo que se conoce como capa Bottleneck. El objetivo de esta capa es comprimir la información obtenida por la red y poder representar la información aprendida por las capas anteriores. La DNN entrenada se utiliza para extraer una nueva representación trama a trama de la señal de entrada al propagar las características originales a través de la DNN y tomar las activaciones de la capa Bottleneck. La DNN utilizada sigue la estructura que se muestra en la Figura 3.8.

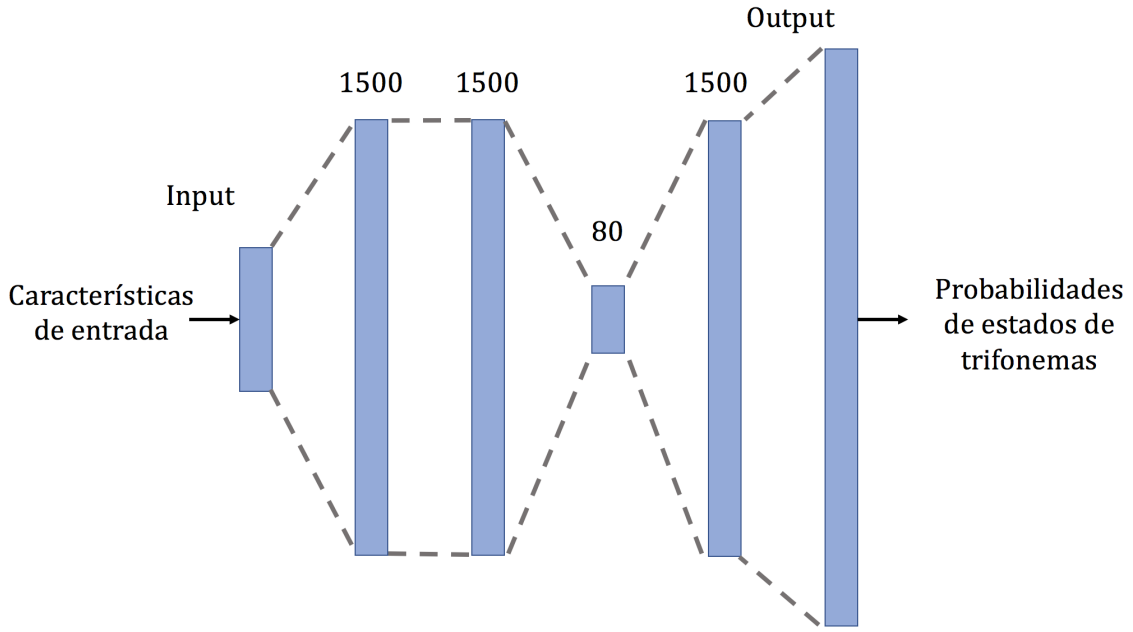

Figura 3.8: Arquitectura de la red neuronal empleada en los experimentos - Topología Bottleneck [22].

La capa de 80 unidades es la capa Bottleneck lineal, mientras que las otras tres aplican la función sigmoide como la función de activación. La capa de salida aplica una función *softmax* y consta de 8832 unidades correspondientes a las distintas clases existentes. Se ha elegido esta topología ya que en los experimentos realizados en [22] se obtuvo una mejora con respecto al sistema base al utilizar las características Bottleneck de una DNN entrenada con características MFCC.

Para entrenar la red neuronal, es preciso realizar un preprocesamiento de los datos. Las matrices de características están preprocesadas de la siguiente manera: ± 15 frames de contexto, se aplica una ventana Hamming seguida de la DCT y se proyecta a los 6 primeros valores. Adicionalmente, se ha realizado una normalización de las características MFCC calculando la

media y la desviación estándar sobre el conjunto entero para facilitar el entrenamiento. La red neuronal lleva a cabo el paso "'forward" (envía su entrada a través de sus capas hacia la salida) y nos quedamos con las activaciones Bottleneck.

4

Análisis de resultados

En el presente capítulo se analizan e interpretan los resultados obtenidos de los distintos experimentos y se lleva a cabo una discusión de dichos resultados. En primer lugar, se exponen los resultados del sistema basado en características acústicas MFCC y, seguidamente, los resultados obtenidos del sistema basado en características Bottleneck.

4.1. Sistema basado en MFCC

4.1.1. Modelo basado en monofonemas y trifonemas ($\Delta + \Delta\Delta$)

En primer lugar, se realiza el entrenamiento del modelo basado en monofonemas que incluye información temporal de los MFCCs, su primera y segunda derivada (Δ+$\Delta\Delta$) teniendo en cuenta $\pm$2 frames. Seguidamente, se lleva a cabo el entrenamiento del modelo basado en trifonemas (tri1 y tri2) que también usan los coeficientes delta. Estudios anteriores apuntan que la introducción de estas características produce mejoras notables en el reconocimiento [10][11]. Basándonos en los resultados obtenidos en [17], se ha definido el número de distribuciones Gaussianas a 30000 y 3000 clases diferentes de trifonemas. En la Tabla 4.1 se exponen los resultados experimentales de dichos entrenamientos utilizando $\Delta + \Delta\Delta$.

	WER [%]
mono ($\Delta + \Delta\Delta$)	85,61
tri1 ($\Delta + \Delta\Delta$)	44,11
tri2 ($\Delta + \Delta\Delta$)	39,76

Tabla 4.1: Word Error Rate resultante del modelo basado en monofonemas y trifonemas ($\Delta + \Delta\Delta$).

Se obtiene una tasa de error de palabra (WER) bastante elevada en el entrenamiento de monofonemas ya que no se tiene en cuenta el contexto fonético. Para reducir dicha tasa, se entrenan los modelos basados en trifonemas con una mayor complejidad que presentan mejoras

significativas. Como se puede observar, se reduce en un 41,5 % la tasa de error de palabra con respecto al modelo basado en monofonemas, es decir, la mejora al utilizar información contextual es notable. Además, el resultado del entrenamiento del modelo de trifonemas tri2 reduce en casi un 5 % el WER con respecto al entrenamiento de trifonemas tri1. Al realizarse cada entrenamiento de forma sucesiva y basándose en el anterior, el nuevo entrenamiento de trifonemas proporciona una mejor tasa de acierto con respecto al entrenamiento del modelo proveniente de los alineamientos utilizando fonemas.

4.1.2. Modelo basado en LDA+MLLT

El siguiente paso es aplicar las técnicas Linear Discriminant Analysis y Maximum Likelihood Linear Transform (LDA+MLLT) sobre el modelo basado en trifonemas. Se trata de una transformación independiente de locutor y proyecta el espacio de características original en un espacio de dimensión inferior. Adicionalmente, se ha demostrado que la transformación LDA funciona mejor cuando se aplica posteriormente una diagonalización MLLT [35]. Como se puede observar en la Tabla 4.2, este resultado presenta una mejora de casi el 6 % sobre el entrenamiento anterior. Esa mejora se debe a que, como ya se ha explicado, estas transformaciones de características son "discriminatorias" e intentan mejorar la separabilidad de las clases acústicas en el espacio de características.

	WER [%]
tri3 (LDA+MLLT)	34,23

Tabla 4.2: Word Error Rate resultante del modelo basado en LDA+MLLT.

4.1.3. Modelo basado en SAT

Partiendo de la alineación obtenida con el modelo tri3, se obtienen nuevos modelos realizando una etapa de adaptación del locutor (Speaker Adaptive Training, SAT) y utilizando el enfoque de regresión lineal de máxima verosimilitud (MLLR). También se obtiene un conjunto de modelos que no aplican este paso de adaptación. El entrenamiento adaptativo es una poderosa técnica de entrenamiento para construir sistemas de reconocimiento de voz en datos no homogéneos. La variabilidad en los datos de entrenamiento puede resultar del cambio del hablante, los diferentes ambientes acústicos o las diferentes condiciones del canal. La idea básica del entrenamiento adaptativo es utilizar una o más transformaciones de características o parámetros del modelo para representar estas diferencias entre el hablante y el entorno.

En primer lugar, se obtiene la decodificación del GMM de trifonemas independiente de locutor (Speaker Independent, SI) y se obtiene una tasa de error de palabra de 40,81 %. Seguidamente, se entrena el GMM aplicando adaptación de locutor (SAT) y utilizando las transformaciones de regresión lineal de máxima verosimilitud del espacio de características y se percibe una mejora del 8 % en la tasa de acierto. Es decir, los resultados experimentales en el contexto de la adaptación de locutor demuestran la efectividad del método propuesto en las tareas de reconocimiento de voz y muestran que se pueden lograr reducciones significativas en la tasa de errores de palabras sobre el paradigma común de interlocutor independiente.

	WER [%]
tri4 (SI)	40,81
tri4 (SAT)	32,04

Tabla 4.3: Word Error Rate resultante del modelo basado en SAT y SI.

4.1.4. Modelo basado en MMI y fMMI

Para las secuencias de observaciones de entrenamiento $\{O_1, ..., O_r, ...O_R\}$ con las transcripciones correspondientes $\{s_r\}$, se puede escribir la función objetivo MMI para el conjunto de parámetros λ, incluido el efecto de escalar las probabilidades acústicas y del modelo de lenguaje según la siguiente expresión:

$$F_{MMI}(\lambda) = \sum_{r=1}^{R} log \frac{p_\lambda(O_r|Ms_r)^K P(s_r)}{\sum_s p_\lambda(O_r|Ms)^K P(s)}$$

donde M_s es el modelo correspondiente a la secuencia de palabras s, $P(s)$ es la probabilidad de esta secuencia determinada por el modelo de lenguaje y k es un factor de escala [9].

MMI maximiza la probabilidad *a posteriori* de las oraciones correctas y para la optimización de la función objetivo utiliza las ecuaciones de actualización de Baum-Welch extendidas (EBW) [26]. El siguiente esquema de entrenamiento discriminatorio es fMMI que aplica la técnica MMI en un espacio de características [26]. En la Tabla 4.4 se exponen los resultados obtenidos en cada iteración en ambos entrenamientos. Se ha utilizado una configuración óptima empleada en otros estudios que realizan 4 iteraciones en el entrenamiento basado en MMI y, para el entrenamiento fMMI no se ha llevado a cabo una fase de alineamiento y se reutiliza el entrenamiento MMI (5 iteraciones) [24]. La tasa de error en el entrenamiento MMI se reduce en cada una de las iteraciones ya que se va ajustando a los datos y disminuye en un 5 % respecto al entrenamiento basado en SAT. Por otro lado, el entrenamiento discriminatorio en el espacio de características fMMI no brinda una mejora con respecto al entrenamiento MMI en este escenario. Se puede observar en la Tabla 4.4 que al aumentar las iteraciones, la tasa de error disminuye pero en la quinta iteración se aleja de la tasa de error mínima.

	WER [%]
tri4_mmi_b0.1 (MMI) - iter 1	29,98
tri4_mmi_b0.1 (MMI) - iter 2	28,61
tri4_mmi_b0.1 (MMI) - iter 3	27,93
tri4_mmi_b0.1 (MMI) - iter 4	27,54
tri4_fmmi_b0.1 (fMMI) - iter 1	31,63
tri4_fmmi_b0.1 (fMMI) - iter 2	29,79
tri4_fmmi_b0.1 (fMMI) - iter 3	29,37
tri4_fmmi_b0.1 (fMMI) - iter 4	29,18
tri4_fmmi_b0.1 (fMMI) - iter 5	29,52

Tabla 4.4: Word Error Rate resultante del modelo basado en MMI y fMMI.

4.1.5. Sistema HMM-DNN (híbrido)

Se ha llevado a cabo un experimento utilizando un sistema híbrido HMM-DNN en el que la DNN reemplaza al GMM. Las características de entrada a la red neuronal son las mismas características adaptadas y procesadas que se pueden incorporar a un modelo basado en GMM en reconocimiento de voz: MFCC ($\Delta + \Delta\Delta$) + LDA + MLLT + fMLLR (características de 40 dimensiones). Se añade a dichas características $\pm$ 4 frames de contexto y se usan como entrada a la red. Se inicializa la red neuronal con una sola capa oculta y, más adelante, se va aumentando el número de capas ocultas en el entrenamiento hasta un total de 5 capas intermedias. En las capas ocultas se utiliza *p-norm*. La expresión que define esta función es la siguiente.

$$y = ||x||_p = \left(\sum_i |x_i|^p \right)^{1/p}$$

El valor de p es configurable pero en los experimentos que se han llevado a cabo [37], se define p = 2 para obtener resultados más óptimos. La salida es una capa softmax con una dimensión igual al número de clases (en este caso, 8832). Esta función calcula las probabilidades de cada clase objetivo sobre todos los estados de trifonemas. La principal ventaja de usar softmax es que el rango de probabilidades de salida es de 0 a 1 y la suma de todas las probabilidades es igual a la unidad. Esta función se utiliza para tareas de clasificación multi-clases y se describe mediante la siguiente expresión [37].

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

En la Tabla 4.5 se exponen los parámetros de la red neuronal implementada.

Nº total de capas	7
Nº capas ocultas	5
Nº neuronas capa de entrada	48
Nº neuronas capas ocultas	500
Nº neuronas capa de salida	8832
Nº épocas de entrenamiento	15
Optimizador	SGD

Tabla 4.5: Parámetros de la red neuronal implementada en Kaldi.

La precisión que alcanza la DNN empleando el conjunto de entrenamiento es **55.62 %** y con el conjunto de validación es **52.15 %**. La implementación del sistema híbrido supera significativamente las configuraciones anteriores ya que alcanza una tasa de error de palabra de 24.02 % (Tabla 4.6).

	WER [%]
DNN híbrida	24,02

Tabla 4.6: Word Error Rate resultante utilizando un sistema híbrido (HMM-DNN) empleando MFCCs.

4.1.6. Análisis comparativo - Sistemas basados en MFCCS

En la Figura 4.1 se ilustran los resultados obtenidos de la tasa de error de palabra en cada uno de los entrenamientos de los modelos incluyendo el experimento del sistema híbrido HMM-DNN. Como se puede observar, al aplicar las transformaciones va mejorando considerablemente la tasa de acierto. Si se analizan los experimentos en los que se han empleado HMM con distintas transformadas y utilizando los coeficientes MFCC (exceptuando la implementación de la DNN), se ha obtenido mejores resultados para el entrenamiento basado en MMI. Por consiguiente, se utilizan las etiquetas o pdf obtenidas en este entrenamiento en la implementación de redes neuronales para la extracción de características Bottleneck. Adicionalmente, se aprecia que la implementación del sistema híbrido con DNNs proporciona la mejor tasa de error de palabra y, por consiguiente, queda reflejada que la utilización de DNNs en reconocimiento de voz lleva a un mejor rendimiento del sistema.

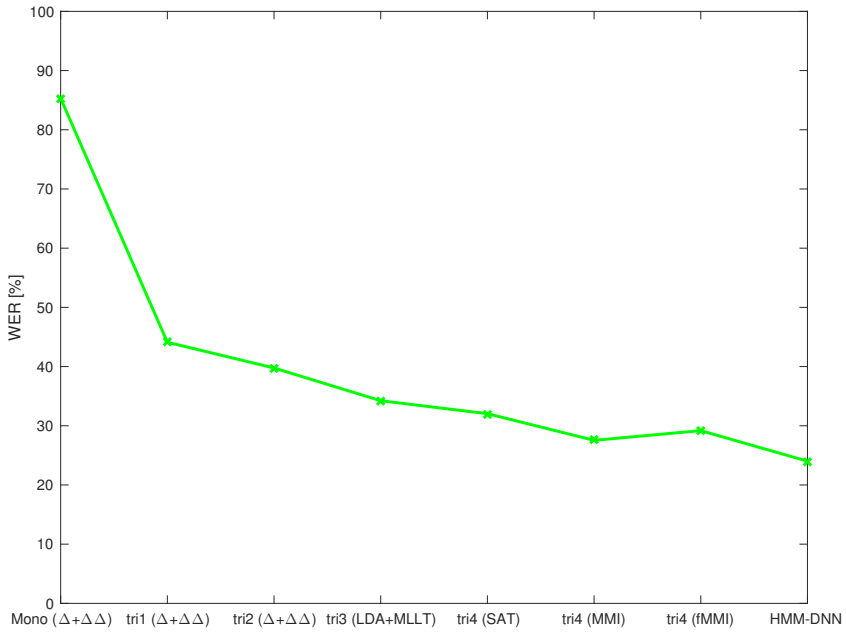

Figura 4.1: Representación gráfica de la tasa de error de palabra obtenida para cada uno de los entrenamientos del sistema basado en características MFCC.

4.2. Sistema basado en características Bottleneck

Una vez se ha conseguido mejorar el sistema base con las transformadas explicadas en las secciones previas, el siguiente paso es entrenar la red neuronal con las características MFCC y extraer una nueva representación de la señal basada en características Bottleneck. Esta DNN se desarrolla empleando la librería Keras.

Como se ha descrito en el Capítulo 3, la arquitectura de la DNN consiste en una capa de entrada, cuatro capas ocultas, una de ellas relativamente pequeña respecto a las otras (capa Bottleneck), y una capa de salida final. Realizamos un preprocesamiento de los datos para

incluir contexto puesto que usamos una DNN y no una red recurrente (explotan las secuencias temporales). El optimizador seleccionado es *adam* que ajusta la tasa de aprendizaje a lo largo del entrenamiento y es una extensión del descenso de gradiente estocástico (Stochastic gradient descent, SGD). El *learning rate* es un parámetro que controla cuánto estamos ajustando los pesos de nuestra red y una tasa de aprendizaje más pequeña puede llevar a pesos más precisos (hasta cierto punto), pero requiere más tiempo para calcular dichos pesos. Por otro lado, se ha definido como función de coste la entropía cruzada que es la función coste que se utiliza para los problemas de clasificación de múltiples clases. La entropía cruzada calcula una puntuación que resume la diferencia promedio entre las distribuciones de probabilidad reales y pronosticadas para todas las clases en el problema. La puntuación se minimiza y el valor ideal de entropía cruzada sería 0. Para dos variables aleatorias discretas p y q, la entropía cruzada se define mediante la siguiente expresión:

$$H(p,q) = -\sum_{x} p(x) log(q(x))$$

La función de activación de las capas ocultas es la función sigmoide exceptuando la capa Bottleneck (lineal). La función sigmoide es una versión suavizada de una función escalonada. Esto es una ventaja, puesto que la función es continua y diferenciable (Figura 4.2).

$$f(x) = \frac{1}{1 + e^{-k(x-x_0)}}$$

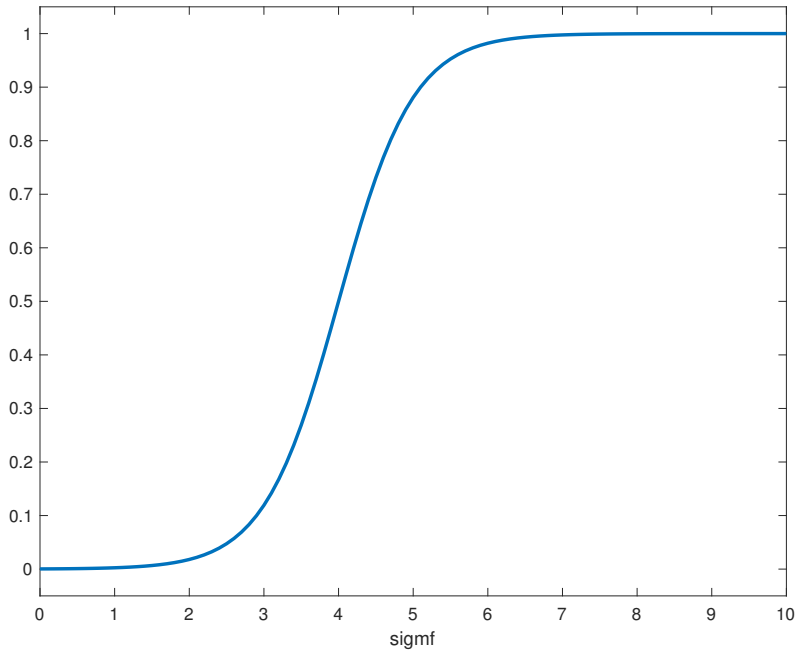

Figura 4.2: Representación gráfica de la función sigmoide.

En la Tabla 4.7 se recogen los parámetros utilizados en el entrenamiento de la DNN.

Nº GPU	1
Nº total de capas	6
Nº capas ocultas	4
Nº neuronas capa de entrada	78
Nº neuronas capas ocultas	1500-1500-80-1500
Nº neuronas capa de salida	8832
Nº épocas de entrenamiento	700
Optimizador	Adam
Función de coste	Cross-entropy

Tabla 4.7: Parámetros de la red neuronal implementada en Keras.

La DNN alcanza una precisión de **49,26 %** con el conjunto de entrenamiento y **48,35 %** con el conjunto de validación (a nivel de frame y entre 8832 clases) y ha sido entrenada usando una GPU. A continuación, en la Tabla 4.8 y en la Figura 4.3 se comparan los resultados obtenidos de tasa de error de palabra del sistema de reconocimiento basado en características MFCCs y basado en características Bottleneck.

	Sistema basado en MFCCs WER [%]	**Sistema basado en características BN WER [%]**
mono ($\Delta + \Delta\Delta$)	85,31	85,61
tri1 ($\Delta + \Delta\Delta$)	44,11	58,66
tri2 ($\Delta + \Delta\Delta$)	39,76	53,73
tri3 (LDA+MLLT)	34,23	34,22
tri4 (SAT)	32,04	31,39
tri4_mmi_b0.1 (MMI)	27,54	29,22
tri4_fmmi_b0.1 (fMMI)	29,18	29,04

Tabla 4.8: Comparación tasas de error de palabra entre el sistema basado en en MFCCs y el sistema basado en características Bottleneck.

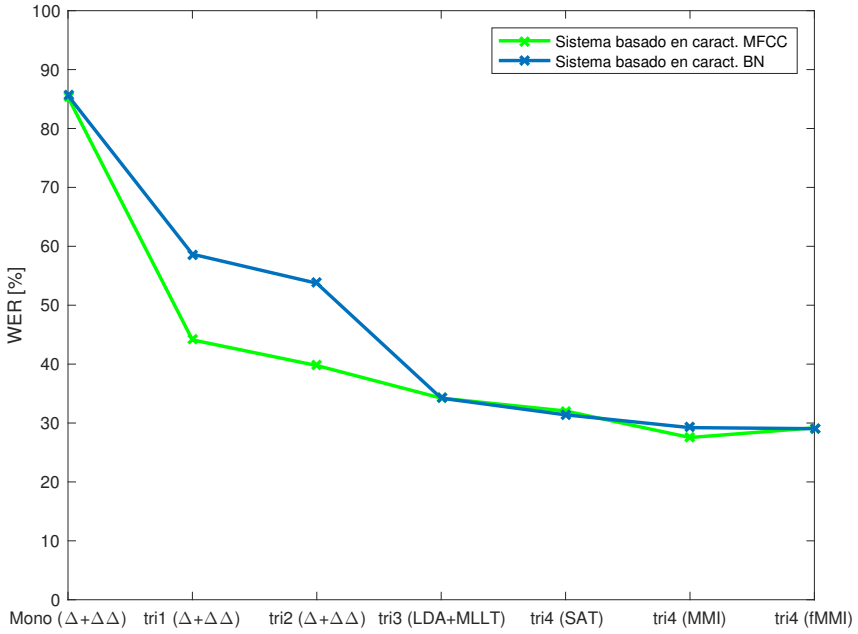

Figura 4.3: Representación gráfica de las tasas de error de palabra del sistema basado en en MFCCs y el sistema basado en características Bottleneck.

De acuerdo con los resultados, las características BN permiten un rendimiento tan bueno como los MFCCs originales en la mayoría de los casos. Se obtienen tasas de error de palabra similares en ambos casos pero algo peores en los entrenamientos de monofonemas y trifonemas (tri1 y tri2). Las características BN ya recogen información del contexto debido al preprocesamiento de los datos de entrada a la red. Con la finalidad de mejorar la tasa de error de palabra en estos modelos se ha llevado a cabo un experimento eliminando los coeficientes delta que reduce la dimensión del espacio de características (Tabla 4.9 y Figura 4.4).

	Sistema basado en caract. BN (sin deltas) - WER [%]
mono	83,97
tri1	42,40
tri2	39,13
tri3 (LDA+MLLT)	34,53
tri4 (SAT)	31,45
tri4_mmi_b0.1 (MMI)	31,43
tri4_fmmi_b0.1 (fMMI)	30,03

Tabla 4.9: Tasas de error de palabra del sistema basado en característica Bottleneck eliminando los coeficientes delta.

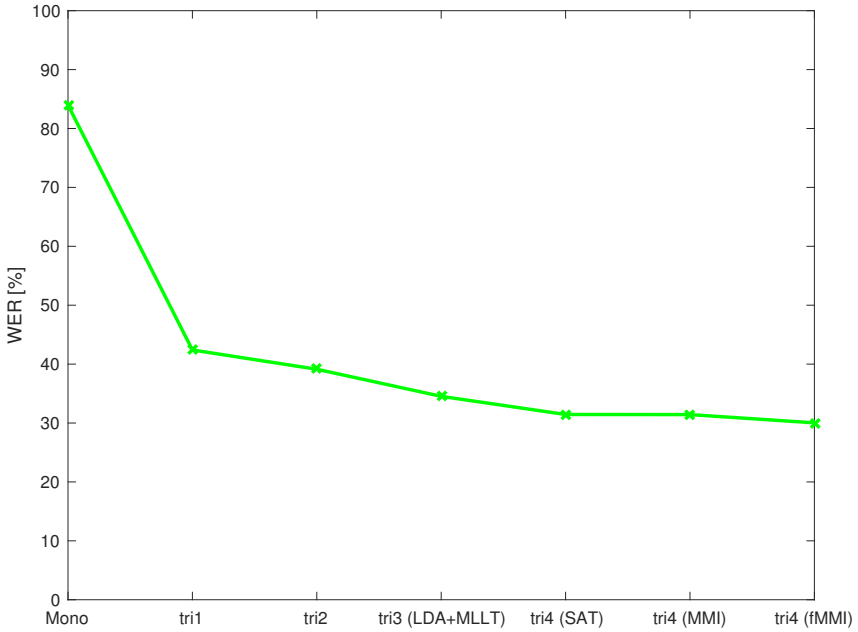

Figura 4.4: Representación gráfica de la tasa de error de palabra del sistema basado en características Bottleneck eliminando los coeficientes delta.

Como se puede observar, se reduce el WER de los modelos mono, tri1 y tri2. Aunque estudios anteriores apuntaban que la introducción de estas características produce mejoras notables en el reconocimiento, estos primeros modelos cuya complejidad es inferior, podrían no ser capaces de lidiar con características en alta dimensionalidad. Por consiguiente, en este escenario agregar los coeficientes delta no aumentan o incluso disminuyen el rendimiento. Para reducir la tasa de error de palabra se utiliza el sistema híbrido HMM-DNN con la misma configuración descrita para los MFCCs (Tabla 4.7). En este caso, la precisión que alcanza la DNN empleando el conjunto de entrenamiento es **64.85 %** y con el subconjunto de validación es **56.73 %**. La implementación del sistema híbrido también supera significativamente las configuraciones anteriores puesto alcanza un WER de 24.98 %, ligeramente superior al valor obtenido para el sistema basado en MFCCs.

	WER [%]
DNN híbrida	24,98

Tabla 4.10: Word Error Rate resultante utilizando un sistema híbrido (HMM-DNN) empleando características Bottleneck.

Basándonos en estudios anteriores [33], la concatenación de las características BN y MFCC, permitiría disminuir el WER drásticamente para todas las condiciones experimentales en comparación con las características MFCC y Bottleneck independientes. Se intentó llevar a cabo este experimento pero por limitación de memoria no fue posible obtener tasas de reconocimiento finales.

5

Conclusiones y líneas futuras

Con el objetivo de analizar si se han cumplido los objetivos establecidos, se realiza un resumen del trabajo desarrollado y una valoración general del mismo. Se exponen las competencias adquiridas y se proponen unas líneas de trabajo futuras.

En primer lugar, se dedicó un periodo de tiempo a la familiarización de las herramientas Kaldi y Keras y al estudio de los conceptos teóricos necesarios para la implementación del sistema de reconocimiento de voz. Para tal estudio, se realizó una búsqueda bibliográfica de materiales y artículos científicos. Se trata de una etapa importante ya que supone adquirir la base teórica necesaria para poder comprender el desarrollo experimental llevado a cabo.

A continuación, aplicando los conocimientos, se desarrolló un sistema base de reconocimiento de voz basado en características acústicas MFCC adaptando una de las recetas de Kaldi al entorno de trabajo. Se llevó a cabo la extracción de características y el entrenamiento del sistema utilizando distintas transformadas y un sistema híbrido HMM-DNN. Seguidamente, se ha implementado una red neuronal profunda en Keras con el objetivo de utilizarla como extractor de características Bottleneck. Dicha red ha sido entrenada y se ha utilizado posteriormente para obtener la nueva representación de la señal basada en Bottlenecks que reemplazarán a los MFCCs del sistema anterior.

Finalmente, el análisis de resultados es el proceso decisivo para valorar el funcionamiento del sistema y analizar si los objetivos establecidos en el inicio del proyecto se han cumplido satisfactoriamente. Se han analizado los resultados obtenidos en los distintos experimentos que se han llevado a cabo para el sistema basado en características MFCC y para el sistema basado en características BN. Posteriormente, se exponen las conclusiones inferidas de dichos resultados.

Tras el resumen descrito, se exponen las aportaciones del proyecto:

- Herramientas del estado del arte. Kaldi representa actualmente una de las herramientas ASR más populares para desarrollar reconocedores de voz. Muchos sistemas de reconocimiento del habla industriales utilizan Kaldi, agregando sus propios datos y modificaciones

al reconocedor y ajustando el modelo. Además de ser ampliamente utilizada en el sector industrial, se emplea en proyectos de investigación de reconocimiento de locutor, reconocimiento de idioma y reconocimiento automático de voz, entre otras [36]. Por otro lado, para la extracción de características BN se ha utilizado la librería Keras de Python que es una de las librerías más poderosas actualmente para desarrollar y evaluar modelos de aprendizaje profundo.

- Análisis comparativo de los resultados del sistema base utilizando MFCCs. Se ha llevado a cabo una comparación de los resultados obtenidos de WER del sistema base utilizando distintas transformadas que permiten mejorar el rendimiento del mismo. Se han entrenado modelos basados en monofonemas ($\Delta + \Delta\Delta$), trifonemas ($\Delta + \Delta\Delta$), LDA + MLLT, SAT, MMI y fMMI. De acuerdo a los resultados obtenidos, es conveniente emplear modelos basados en trifonemas ya que presentan una menor probabilidad de error con respecto a los modelos basados en monofonemas. Se ha realizado cada entrenamiento de forma sucesiva, basándose en el anterior y el sistema que proporciona mejores tasas de error de palabra es el sistema basado en la técnica MMI. Adicionalmente, se ha realizado un experimento utilizando un sistema híbrido en el que los GMMs se sustituyen por una DNN en el sistema HMM, una de las técnicas en el estado del arte en reconocimiento de voz, que permite optimizar el rendimiento del sistema con respecto a las configuraciones anteriores.

- Estudio de la utilización de características Bottleneck en un sistema de reconocimiento de voz. Se han comparado los resultados obtenidos en el sistema de reconocimiento de voz basado en una parametrización clásica de la señal MFCC y el sistema de reconocimiento de voz basado en características Bottleneck. En este escenario, las características BN permiten un rendimiento tan bueno como los MFCC originales excepto en los entrenamientos con modelos cuya complejidad es inferior ya que podrían no lidiar con características en alta dimensionalidad. Es decir, en este contexto agregar los coeficientes delta a las características BN disminuye el rendimiento del sistema desarrollado y, a pesar de que en otros trabajos los BN proporcionan un mejor rendimiento que los MFCCs [22], no se obtiene una mejora significativa en las tasas de reconocimiento con la utilización de dichas características.

Este trabajo ha significado una toma de contacto con el mundo de la investigación y me ha permitido conocer y aprender herramientas útiles en reconocimiento automático de voz. Se han involucrado conocimientos adquiridos en el Master y se ha profundizado en el mundo del ASR además de desarrollar competencias profesionales.

En cuanto al trabajo futuro, uno de los puntos pendientes sería investigar en profundidad la concatenación de características MFCC y Bottleneck como entrada al sistema de reconocimiento de voz y analizar el rendimiento del sistema para evaluar si permite una mejora en las tasas de reconocimiento en un entorno experimental distinto. Además, se explorarían las características Bottleneck que se utilizan en otros estudios para así poder proporcionar mejores resultados. Por otro lado, se propone variar los parámetros correspondientes a las redes neuronales (learning rate, número de capas, número de neuronas, etc) con la finalidad de conseguir un mayor rendimiento del sistema.

Glosario de acrónimos

- **ASR**: Automatic Speech Recognition
- **MFCC**: Mel Frequency Cepstral Coefficients
- **AI**: Artificial Intelligence
- **ML**: Machine Learning
- **DL**: Deep Learning
- **DNN**: Deep Neural Network
- **LVCSR**: Large Vocabulary Continuos Speech Recognition
- **HMM**: Hidden Markov Models
- **GMM**: Gaussian Mixture Model
- **DFT**: Discrete Fourier Transform
- **DCT**: Discrete Cosine Transform
- **CMVN**: Cepstral Mean and Variance Normalization
- **ANN**: Artificial Neural Network
- **LDC**: Linguistic Data Consortium
- **FST**: Finite State Transducer
- **LDA**: LinearDiscriminant Analysis
- **MLLT**: Maximum Likelihood Linear Transform
- **SAT**: Speaker Adaptive Training
- **MMI**: Maximum Mutual Information
- **fMMI**: featured Maximum Mutual Information
- **WER**: Word Error Rate
- **GPU**: Graphics Processing Unit
- **DT**: Decision Tree
- **EBW**: Extended Baum-Welch
- **SI**: Speaker Independent
- **SGD**:Stochastic gradient descent

Bibliografía

[1] MATLAB and Statistics Toolbox Release 2016a. The Mathworks, Inc., Natick, Massachusetts, United States.

[2] S Balakrishnama and A Ganapathiraju. Linear Discriminant Analysis - A Brief Tutorial. *Institute for Signal and Information Processing, Mississippi State University*, 1998.

[3] Douglas D. O'Shaughnessy. Invited paper: Automatic speech recognition: History, methods and challenges. *Pattern Recognition*, 41:2965–2979, 10 2008.

[4] Joseba Elola. El cerebro artificial que piensa por ti. *El País*, 2017.

[5] J. Godfrey and E. Holliman. Switchboard-1 release 2 LDC97S62. Philadelphia: Linguistic Data Consortium, 1993.

[6] R. E. Gruhn, W. Minker, and S. Nakamur. *Statistical Pronunciation Modeling for Non-Native Speech Processing*. Springer Berlin Heidelberg, 2011.

[7] F. Grézl, M. Karafiát, and L. Burget. Investigation into Bottleneck features for meeting speech recognition. *Proc. Interspeech*, 2009.

[8] B H. Juang and Lawrence Rabiner. Automatic speech recognition - a brief history of the technology development. 01 2005.

[9] T. Hain, P. C. Woodland, G. Evermann, M. J. F. Gales, Xunying Liu, G. L. Moore, D. Povey, and Lan Wang. Automatic transcription of conversational telephone speech. *IEEE Transactions on Speech and Audio Processing*, 13(6):1173–1185, Nov 2005.

[10] C. Hanilci and F. Ertas. Impact of voice excitation features on speaker verification. In *2011 7th International Conference on Electrical and Electronics Engineering (ELECO)*, pages II–157–II–160, Dec 2011.

[11] B. A. Hanson and T. H. Applebaum. Robust speaker-independent word recognition using static, dynamic and acceleration features: experiments with Lombard and noisy speech. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 857–860 vol.2, April 1990.

[12] J. Hilera and V. Martínez Hernando. Redes neuronales artificiales : fundamentos, modelos y aplicaciones. May, 2019.

[13] Roger Hsiao and Tanja Schultz. Generalized discriminative feature transformation for speech recognition. pages 664–667, 01 2009.

[14] Qi Jun, Dong Wang, and Javier Tejedor. Subspace models for bottleneck features. 08 2013.

[15] Alexey Karpov, Rodmonga Potapova, and Iosif Mporas. *Speech and Computer: 19th International Conference, SPECOM 2017, Hatfield, UK, September 12-16, 2017, Proceedings*. 01 2017.

[16] Irina Kipyatkova and Alexey Karpov. DNN-Based Acoustic Modeling for Russian Speech Recognition using Kaldi. pages 246–253, 08 2016.

[17] Piotr Kozierski, Talar Sadalla, Szymon Drgas, Adam Dabrowski, Joanna Zietkiewicz, and Wojciech Giernacki. Acoustic model training, using kaldi, for automatic whispery speech recognition. pages 109–114, 09 2018.

[18] Katri Leino. Breakthroughs in Automatic Speech Recognition Technology. July 2015.

[19] J. Levis and R. Suvorov. Encyclopedia of applied linguistics, automatic speech recognition. 2012.

[20] Dong Yu Li Deng. *Deep Learning: Methods and Applications*. Foundations and Trends® in Signal Processing, 2013.

[21] A. Lozano-Diez, R. Zazo, D. T. Toledano, and J. Gonzalez-Rodriguez. An analysis of the influence of deep neural network (DNN) topology in bottleneck feature based language recognition. 2017.

[22] A. Lozano-Díez, A. Silnova, P. Matejka, O. Glembek, O. Plchot, J. Pesan, L. Burget, and J. González-Rodríguez. Analysis and Optimization of Bottleneck features for speaker recognition. *Proc. of Odyssey*, June 2016.

[23] G. A. Martínez Mascorro and G. Aguilar Torres. Reconocimiento de voz basado en MFCC, SBC y espectrogramas. *INGENIUS*, 2013.

[24] B. Popović, S. Ostrogonac, E. Pakoci, N. Jakovljevic, and V. Delić. Deep neural network based continuous speech recognition for serbian using the kaldi toolkit. *Lecture Notes in Computer Science*, 9319:186–192, 01 2015.

[25] D. Povey, A. Ghoshal, G. Boulianne, L Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, 2011. IEEE Catalog No.: CFP11SRW-USB.

[26] Daniel Povey, Dimitri Kanevsky, Brian Kingsbury, Bhuvana Ramabhadran, George Saon, and Karthik Visweswariah. Boosted mmi for model and feature-space discriminative training. pages 4057–4060, 03 2008.

[27] Daniel Povey, Xiaohui Zhang, and Sanjeev Khudanpur. Parallel training of deep neural networks with natural gradient and parameter averaging. *CoRR*, abs/1410.7455, 2014.

[28] R. Prieto, A. Herrera, J. L. Pérez, and A. Padrón. El modelo neuronal de Mccullock y Pitts, Interpretación comparativa del modelo.

[29] L. R. Rabiner and R. W. Schafer. *Introduction to Digital Speech Processing*. Foundations and Trends® in Signal Processing, 2007.

[30] M. Ravanelli, T. Parcollet, and Y Bengio. The Pytorch-Kaldi speech recognition toolkit, 11 2018.

[31] Sebastian Ruder. An overview of gradient descent optimization algorithms. 09 2016.

[32] A. Serrano, E. Soria, and J. Martín. Redes neuronales artificiales - Doctorado. 2009-2010.

[33] Doroteo T. Toledano, Alfonso Ortega, Antnio Teixeira, Joaquin Gonzalez-Rodriguez, Luis Hernandez-Gomez, Ruben San-Segundo, and Daniel Ramos. *Advances in Speech and Language Technologies for Iberian Languages: IberSPEECH 2012 Conference, Madrid, Spain, November 21-23, 2012. Proceedings*. Springer Publishing Company, Incorporated, 2012.

[34] Jordi Torres. *Deep Learning, Introducción práctica con Keras.* Foundations and Trends® in Signal Processing, 2018.

[35] P. Upadhyaya, S. Mittal, Y. Vardhan, O. Farooq, and R. Abidi. Speaker Adaptive Model for Hindi Speech using Kaldi Speech Recognition toolkit. 02 2018.

[36] R. Yazdani, A. Segura, J. Arnau, and A. Gonzalez. Low-power automatic speech recognition through a mobile gpu and a viterbi accelerator. *IEEE Micro*, 37(1):22–29, Jan 2017.

[37] X. Zhang, J. Trmal, D. Povey, and S. Khudanpur. Improving deep neural network acoustic models using generalized maxout networks. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 215–219, May 2014.